Non-autoregressive Math Word Problem Solver with Unified Tree Structure

EMNLP 2023

Yi Bin[1], Mengqun Han[2], Wenhao Shi[2], Lei Wang[3], Yang Yang[2]
See-Kiong Ng[1], Heng Tao Shen[2]

[1]National University of Singapore
[2]University of Electronic Science and Technology of China
[3]Singapore Management University

Task Description

Math Word Problem Solving

Part 1.

Background

- **Math Word Problem Solving**
  - Answer a mathematical query according to the text description

Math Word Problem Solving

Word Problem
Oceanside Bike Rental Shop charges 17 dollars plus 7 dollars an hour for renting a bike.Tom paid 80 dollars to rent a bike. How many hours did he pay to have the bike checked out?

Equation & Solution

$$17 + (7 \times x) = 80$$

$$x = 9$$

Operators $+, -, \times, \div$

Math23K (Wang et al., 2017)

dataset	# problems	# templates	# sentences	# words	problem types
Alg514	514	28	1.62k	19.3k	algebra, linear
Dolphin1878	1,878	1,183	3.30k	41.4k	number word problems
DRAW-1K	1,000	Unknown	6.23k	81.5k	algebra, linear, one-VAR
Math23K	23,161	2,187	70.1k	822k	algebra, linear, one-VAR

Math Quesition Answering

Passage (some parts shortened)

In **1517, the seventeen-year-old King sailed to Castile**. There, his Flemish court .... **In May 1518, Charles traveled to Barcelona in Aragon**.

Question	Answer	BiDAF
Where did Charles travel to first, Castile or Barcelona?	Castile	Aragon

DROP (Dheeru Dua et al, 2019)

Reasoning	Ratio
Subtraction	28.80%
Comparison	18.20%
Selection	19.40%
Addition	11.70%

Reasoning	Ratio
Count	16.50%
Other Arithmetic	3.20%
Other	6.80%

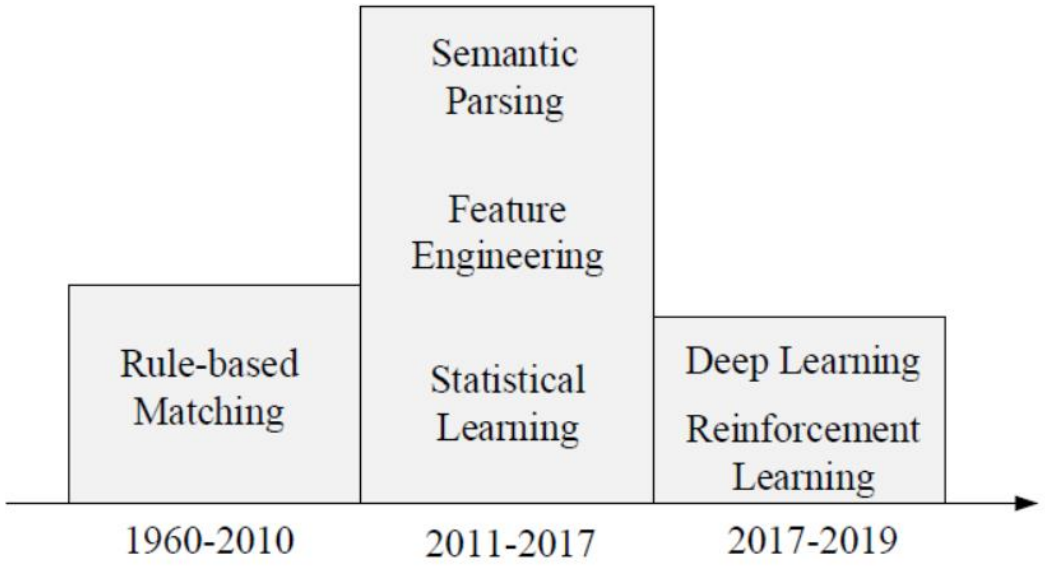
Dheeru Dua et al. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. NAACL. 2019.

Part 1.

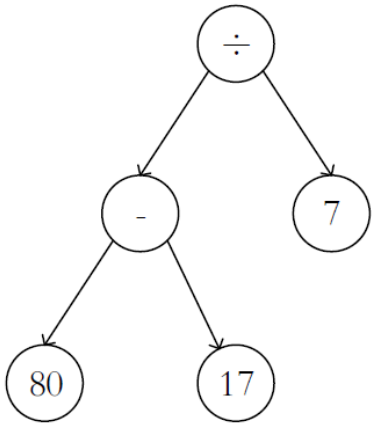
Background

- **Math Word Problem Solving: History**
  - Tree-Based Methods
    - Transform the derivation of the arithmetic expression to constructing an equivalent tree
  - DL-based Methods
    - Learn an effective feature representation in a data-driven manner

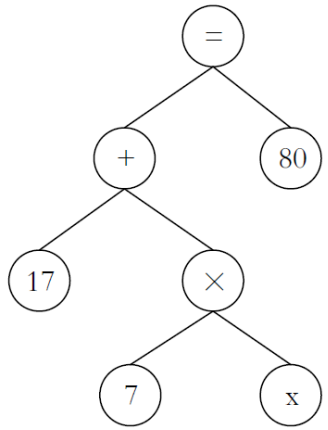
Technology evolving trend in solving MWPs



Expression Tree



Equation Tree



- (Roy et al., 2015): Decision problem for expression tree
  - Use binary SVM classifier for Relevance/LCA Operation

Dongxiang Zhang et al. The Gap of Semantic Parsing: A Survey on Automatic Math Word Problem Solvers. 2018.
Subhro Roy et al. Solving General Arithmetic Word Problems. ACL. 2015.

Part 1.

Background

- **Math Word Problem Solving**
  - Input
    - The text description for the math problem, represented in the form of a sequence of k words $\langle w_0, w_1, \ldots, w_k \rangle$
    - There are n quantities $q1, q2, \ldots, qn$ mentioned in the text and an unknown variable x whose value is to be resolved
  - Goal
    - Extract the relevant quantities and map this problem into an arithmetic expression E whose evaluation value provides the solution to the problem
    - Four types of fundamental operators $O = \{+, -, \times, \div\}$ involved in the expression

Word Problem
Oceanside Bike Rental Shop charges 17 dollars plus 7 dollars an hour for renting a bike.Tom paid 80 dollars to rent a bike. How many hours did he pay to have the bike checked out?

Equation & Solution

$17 + (7 \times x) = 80$

$x = 9$

Operators $+, -, \times, \div$

Dongxiang Zhang et al. The Gap of Semantic Parsing: A Survey on Automatic Math Word Problem Solvers. 2018.

Part 1.

Background

- **Mathematical Reasoning Tasks**
  - ★ **Math Word Problem Solving**
    - **Textual Problem, Multimodal Problem**
  - Theorem Proving
    - Formal Problem, Informal Problem
    - Formal + Informal Problem
  - Geometry Problem Solving
    - Without Annotations, With Annotations
  - Math Question Answering
    - Single Benchmark, Unified Benchmark
  - Other Quantitative Problems
    - Diagram, Finance, Science, Programming

Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.

Background

• Theorem Proving

Informal Statement	Let a, b, c be the sides of a triangle. Prove that $$\frac{\sqrt{b+c-a}}{\sqrt{b}+\sqrt{c}-\sqrt{a}}+\frac{\sqrt{c+a-b}}{\sqrt{c}+\sqrt{a}-\sqrt{b}}+\frac{\sqrt{a+b-c}}{\sqrt{a}+\sqrt{b}-\sqrt{c}} \leq 3$$
Formal Statement	`theorem imosl_2006_algebra_p5 (a b c : ℝ) (h₀ : a > 0 ∧ b > 0 ∧ c > 0) (h₁ : a + b > c ∧ a + c > b ∧ b + c > a) : (real.sqrt (b + c - a) / (real.sqrt b + real.sqrt c - real.sqrt a)) + (real.sqrt (c + a - b) / (real.sqrt c + real.sqrt a - real.sqrt b)) + (real.sqrt (a + b - c) / (real.sqrt a + real.sqrt b - real.sqrt c)) 3 := begin sorry end`

Task Definition

- Generate a proof given a conjecture (the target theorem) and a knowledge base of known facts
- It's all expressed in a formal language

Task Description

- The problem is to demonstrate the truth of a mathematical claim (a theorem)
  through a sequence of logical arguments (a proof)
- Choose effective multi-step strategies, using background knowledge, and performing symbolic manipulations

Mingzhe Wang et al. Learning to Prove Theorems by Learning to Generate Theorems. NeurIPS. 2020.
Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Ziyin Zhang et al. DEEPTHEOREM: Advancing LLM Reasoning for Theorem Proving Through Natural Language and Reinforcement Learning. 2025.

Background

• Theorem Proving

Model	FIMO	
	out.	*proc.*
Claude3.7-Sonnet	34.92	26.28
Gemini2.5-Pro	57.14	54.06
Gemini2.5-Flash	30.16	28.95
GPT-4o	34.92	30.70
o1-mini	60.32	55.23
o1	66.67	61.00
o3-mini	80.95	77.61
Qwen2.5-Inst-7B	30.16	21.13
Qwen2.5-Inst-72B	49.21	37.35
Qwen2.5-Math-Inst-7B	28.57	18.86
Qwen2.5-Math-Inst-72B	47.62	36.02
DS-Prover-v1.5-RL-7B	25.40	13.81
DS-Prover-v2-7B	30.16	21.86
R1-Distill-7B	6.35	4.27
R1-Distill-70B	17.46	14.05
QwQ-32B	17.46	15.41
Llama3.3-Inst-70B	41.27	27.33
*DeepTheorem-RL-7B	55.56	39.07

Outcome Evaluation

- Evaluation framework that leverages multiple entailing and contradictory variants derived from each theorem
- Indirectly estimate its theorem justification ability

Process Evaluation

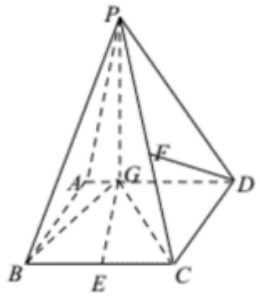
- Evaluates the quality of proof along four dimensions:
  (1) Logical Validity (2) Completeness
  (3) Correctness (4) Clarity
- Use GPT-4o as the LLM judge
- Ask it to score the proof using a weighted sum of the four dimensions

Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Ziyin Zhang et al. DEEPTHEOREM: Advancing LLM Reasoning for Theorem Proving Through Natural Language and Reinforcement Learning. 2025.

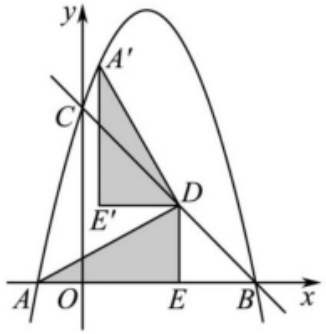
Part 1.

Background

- **Geometry Problem Solving**



ID: prob_60213
Question: In the pentahedron P-ABCD, the base ABCD is a parallelogram, ⋯ ⋯ E is the midpoint of BC. Find the distance from point D to the plane PBG.
Choice:
(A) $\frac{5}{2}$. (B) $\frac{3}{2}$.
(C) $\frac{1}{2}$. (D) $\frac{1}{5}$.
Answer: (B)



ID: prob_60156
Question: As shown in Figure, given the parabola $y = -x^2 + 3x + 4$ intersecting the x axis at A and B (point A is to the left of point B) ⋯ If the area of Δ A D E is S, find the value of m that maximizes S
Choice:
(A) 1.5. (B) 2.5.
(C) 0.5. (D) 3.
Answer: (A)

Task Definition

- Answer a geometric query according to the text description and a diagram

Task Description

- Problem consists of a textual description and a diagram
- The multimodal inputs describe the entities, attributes, and relationships of geometric elements
- The goal is to find the numeric solution to an unknown variable

Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Jiaxin Zhang et al. GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving. ACL Finding. 2024.

Background

• Geometry Problem Solving

Expriemental Result

Model	GeoEval-2000 A (%)	GeoEval-2000 T (%)
CodeGen2-16B ◊	28.76	22.06
GPT-3.5 ◊	24.71	21.27
GPT-4 ◊	27.95	43.86
WizardMath-70B ◊	**55.67**	34.20
WizardMath-7B-V1.1 ◊	54.78	32.76
llava-7B-V1.5	12.80	21.01
Qwen-VL	25.60	25.97
mPLUG-Owl2	37.76	n/a
InstructBLIP †	52.18	n/a
GPT-4V	37.22	**43.86** ‡

Evaluation

◦ Given the model's intention to produce responses
- Precise answer (e.g., "3.15")
- The corresponding option letter (e.g., "A")

◦ Regard a prediction as accurate
if it either matches the golden answer or
the golden option letter

Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Jiaxin Zhang et al. GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving. ACL Finding. 2024.

Part 1.

Background

• Math Question Answering

DROP (Dheeru Dua et al, 2019)

Reasoning	Passage (some parts shortened)
Comparison (18.2%)	In 1517, the seventeen-year-old King sailed to Castile. There, his Flemish court .... In May 1518, Charles traveled to Barcelona in Aragon.

Question	Answer	BiDAF
Where did Charles travel to first, Castile or Barcelona?	Castile	Aragon

Reasoning	Ratio
Subtraction	28.80%
Comparison	18.20%
Selection	19.40%
Addition	11.70%

Reasoning	Ratio
Count	16.50%
Other Arithmetic	3.20%
Other	6.80%

Task Definition

- Answer a math-related question presented in natural language

Task Description

- Question answering (QA) benchmarks that center around mathematical reasoning
- Requires discrete reasoning to answer questions such as *"Which kicker kicked the most field goals?"* over the content of paragraphs

Wentao Liu et al. Mathematical Language Models: A Survey. 2023.
Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Dheeru Dua et al. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. NAACL. 2019.

Background

• Math Question Answering

Expriemental Result

Models	EM	F_1
Specialized Models		
NumNet (Ran et al., 2019)	64.9	68.3
MTMSN (Hu et al., 2019)	76.7	80.5
NeRd (Chen et al., 2020c)	78.6	81.9
NumNet+ (Ran et al., 2019)	81.1	84.4
QDGAT (Chen et al., 2020a)	**84.1**	**87.1**

Models	EM	F_1
Language Models		
T5 (Yoran et al., 2022)	61.8	64.6
GenBERT (Geva et al., 2020)	68.8	72.3
PReasM (Yoran et al., 2022)	69.4	72.3
POET-SQL_{BART}	77.7	80.6
POET-Math+SQL_{BART}	**78.0**	**80.9**

Evaluation

◦ Given the model's intention to produce responses
- Precise answer (e.g., "3.15")
- The corresponding option letter (e.g., "A")

◦ Regard a prediction as accurate if it either matches the golden answer or the golden option letter

Pan Lu et al. A Survey of Deep Learning for Mathematical Reasoning. 2022.
Dheeru Dua et al. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. NAACL. 2019.

Background

• Math Word Problem Solving: Model Comparison

Value accuracy (final answer)

	Model	Test	5-fold
Seq2Seq / Tree	GroupAttn(2019)	69.5	66.9
	GTS (2019)	75.6	74.3
	G2T(2020)	77.4	75.5
	mBERT(2021)	75.1	-
	Symbol-Dec(2021)	-	75.7
	BERTGen(2021)	76.6	-
	PLM-Gen (2021)	76.9	-
	H-Reasoner(2021)	83.9	82.2
	BERT-T(2021a)	84.4	82.3
	Rank†(2021)	85.4	-
	Logic-Dec(2022b)	83.4	-
	T-Dis(2021b)	79.1	77.2
	Prototype (2021)	83.2	-
	Textual-CL (2022a)	85.0	82.6†
	Ana-CL (2022)	85.6	83.2†
Seq2Exp	E-Pointer†(2020)	78.7	76.5
	DAG (2021)	77.5	75.1
	M-Tree(2022a)	82.5	80.8†
	RE-Ext(2022)	85.4	83.3
	M-View♢(2022a)	85.6	83.1
	Elastic†(2022)	84.8	82.9
	MWP-NAS(2023)	84.4	
LLM	gpt-3.5-turbo†(2022)	54.8	-
	Self-Consistency†(2022b)	66.1	-
	Ours **Expression Tree**	**86.2** ±0.30	**84.1** ±0.65
	Ours (Layer-Shared)	**85.6** ±0.25	**83.4** ±0.38

♠ **Seq2Seq, Seq2Tree is generally used terms in research area**

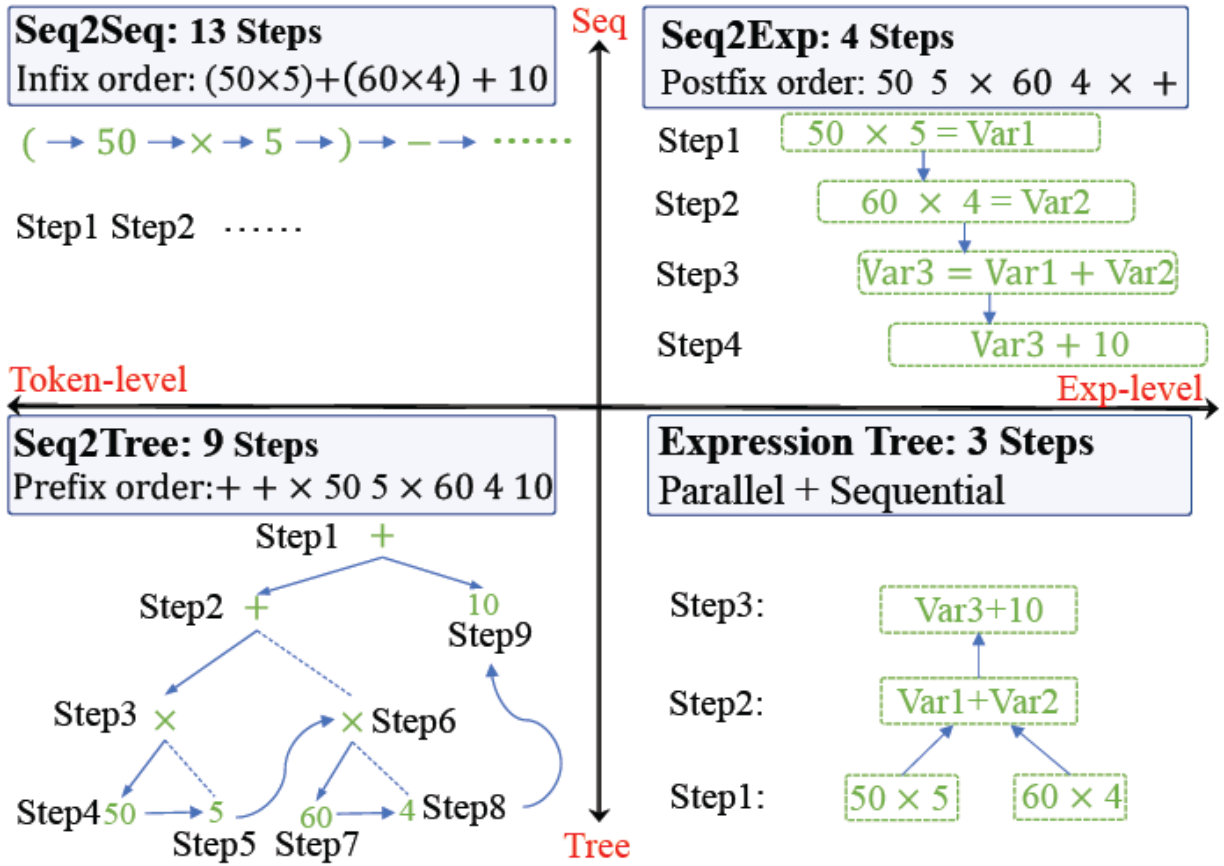
♣ **Seq2exp, ExpressionTree is defined by (Wenqi Zhang et al. 2023)**

Wenqi Zhang et al. An Expression Tree Decoding Strategy for Mathematical Equation Generation. EMNLP. 2023.

Background

• Math Word Problem Solver

Question: Two cars are traveling from two places 10 km apart, the speed of the first car is 50km/h, the speed of the second car is 60km/h, the first car drives for 5 hours, the second car drives for 4 hours, finally how far apart are the two cars?



Seq2seq

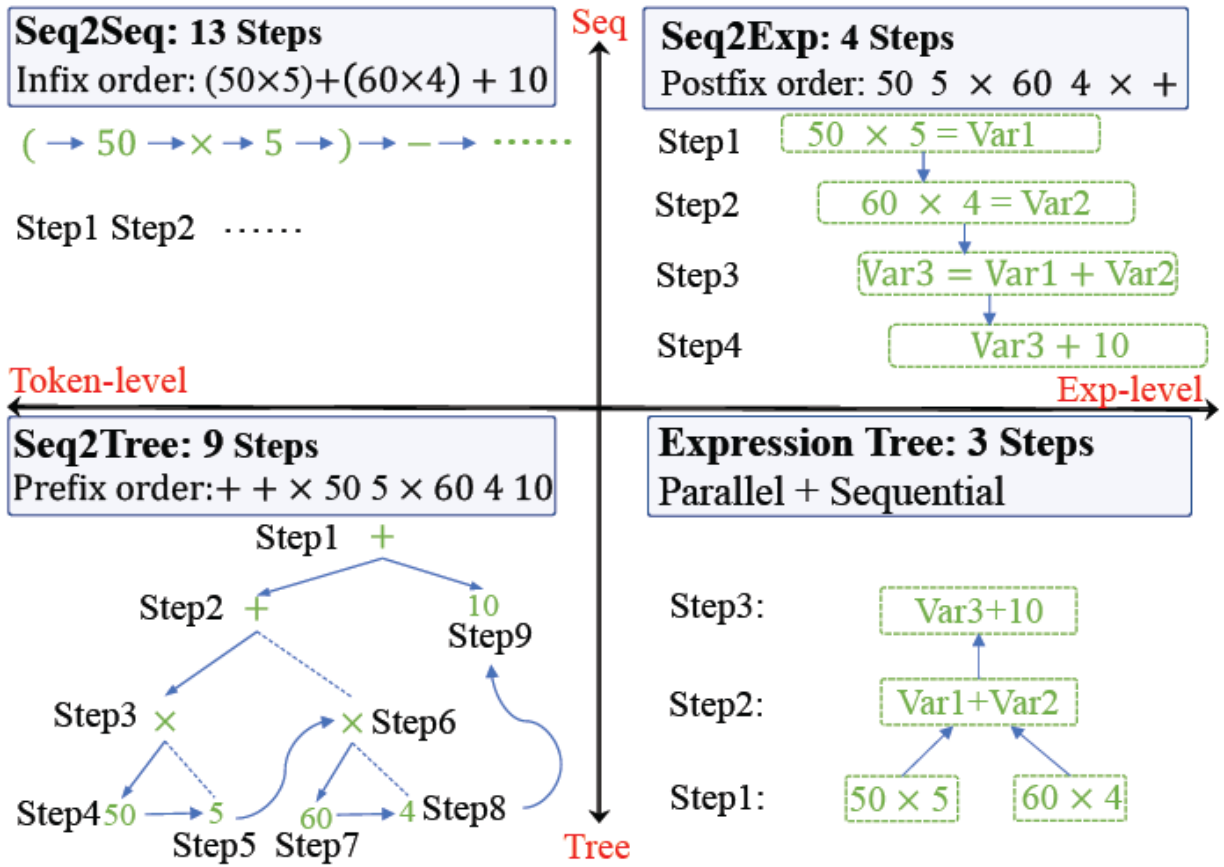
- Consider mathematical symbols as a special kind of language
- Employ sequential generation for the equation

Wenqi Zhang et al. An Expression Tree Decoding Strategy for Mathematical Equation Generation. EMNLP. 2023.

Background

• Math Word Problem Solver

Question: Two cars are traveling from two places 10 km apart, the speed of the first car is 50km/h, the speed of the second car is 60km/h, the first car drives for 5 hours, the second car drives for 4 hours, finally how far apart are the two cars?



Seq2seq

- Consider mathematical symbols as a special kind of language
- Employ sequential generation for the equation

Seq2tree

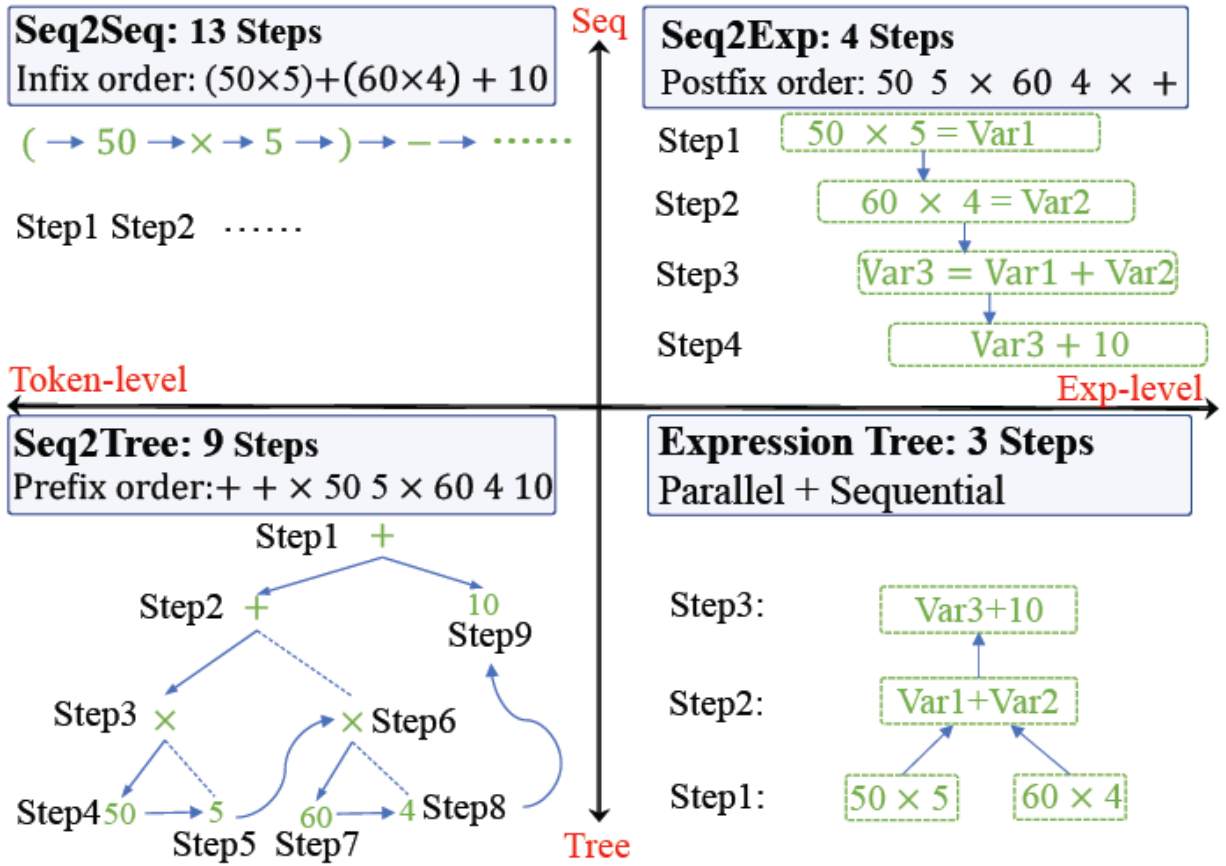
- Consider it as an equation tree generation
- Predict pre-order tokens one by one

Wenqi Zhang et al. An Expression Tree Decoding Strategy for Mathematical Equation Generation. EMNLP. 2023.

Background

• Math Word Problem Solver

Question: Two cars are traveling from two places 10 km apart, the speed of the first car is 50km/h, the speed of the second car is 60km/h, the first car drives for 5 hours, the second car drives for 4 hours, finally how far apart are the two cars?



Wenqi Zhang et al. An Expression Tree Decoding Strategy for Mathematical Equation Generation. EMNLP. 2023.

Seq2seq

- Consider mathematical symbols as a special kind of language
- Employ sequential generation for the equation

Seq2tree

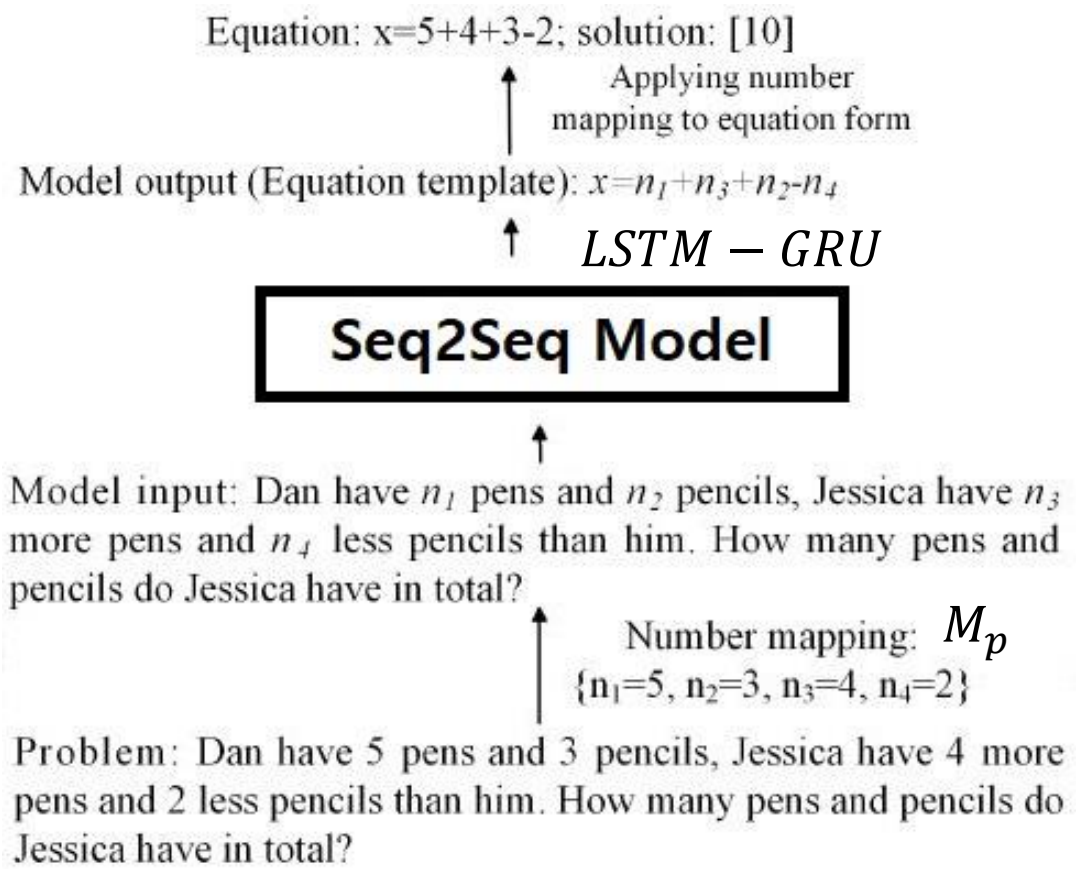
- Consider it as an equation tree generation
- Predict pre-order tokens one by one

Seq2exp

- Mostly place emphasis on generating a mathematical expression step by step
- Each expression represents a problem solving

Background

• Math Word Problem Solver : Seq2Seq



◦ A Seq2Seq mapping problem:
Natural language ➡ Mathematical expression

Sequentially generate the expression

◦ To decrease the diversity of equations
- Map each equation E_p to an equation template T_p through a number mapping M_p $\{n_1, n_2, \ldots, n_m\}$

V_p $V_p = \{v_1, \ldots, v_m, x_1, \ldots, x_k\}$

T_p $x = n_1 + n_3 + n_2 - n_4$

Limitation **Seq2Seq ➡ Seq2Tree**

◦ Ignore the arithmetic properties of the expression

◦ Do not match the goal-driven mechanism in human problem solving

Yan Wang et al. Deep neural solver for math word problems. EMNLP. 2017.

Background

• Math Word Problem Solver : Seq2Seq

Encoder (GRU)

$$z_t = \sigma(W^{(z)}x_t + U^z h_{t-1}) \quad \text{(Update gate)}$$
$$r_t = \sigma(W^{(r)}x_t + U^r h_{t-1}) \quad \text{(Reset gate)}$$
$$\hat{h}_t = tanh(r_t \odot U h_{t-1} + W x_t) \quad \text{(New memory)}$$
$$h_t = (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1} \quad \text{(Hidden state)}$$

The output probability

$$P(\hat{r}_t|h_t) = \frac{\rho_t \odot e^{h_t^T W^s}}{\sum \rho_t \odot e^{h_t^T W^s}}$$

A binary vector p_t can be generated depends on r_{t-1} and these rules

Decoder (LSTM)

$$i_t = \sigma(W^{(i)}x_t + U^i h_{t-1}) \quad \text{(Input gate)}$$
$$f_t = \sigma(W^{(f)}x_t + U^f h_{t-1}) \quad \text{(Forget gate)}$$
$$o_t = \sigma(W^{(o)}x_t + U^o h_{t-1}) \quad \text{(Output gate)}$$
$$\tilde{c}_t = tanh(W^{(c)}x_t + U^{(c)} h_{t-1}) \quad \text{(New memory)}$$
$$c_t = f_t \odot \tilde{c}_{t-1} + i_t \odot \tilde{c}_t \quad \text{(Final memory)}$$
$$h_t = o_t \odot tanh(c_t) \quad \text{(Hidden state)}$$

The five predefined rules

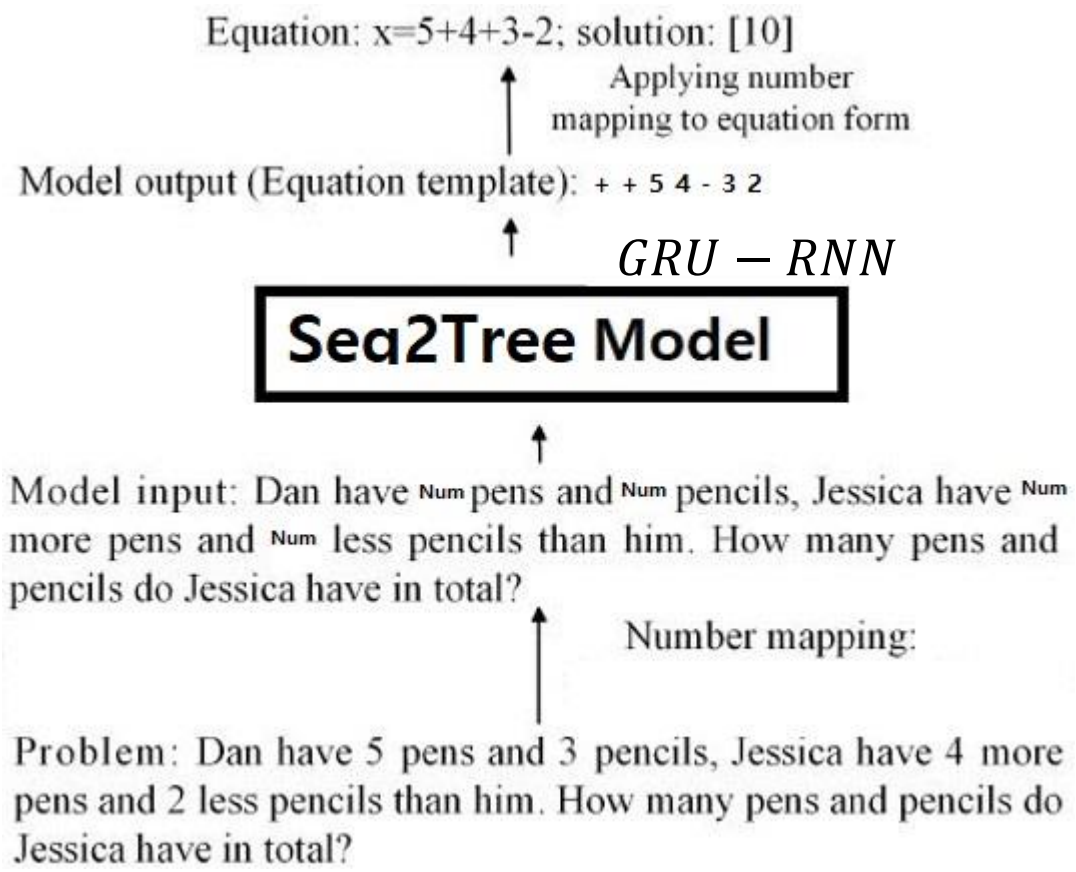
- Rule 1: If r_{t-1} in $\{+, -, *, /\}$, then r_t will not in $\{+, -, *, /,), =\}$;

The rest is omitted.

Yan Wang et al. Deep neural solver for math word problems. EMNLP. 2017.

Background

- **Math Word Problem Solver: Seq2Tree**



- Problem Text P
  - N_p $\{n_1, n_2, \ldots, n_m\}$ denote the ordered list of numeric values in P according to their order in the problem text
- Solution Expression Tree T
  - T can be easily transformed from the solution expression
  - For each token y in the target vocabulary V dec of P, its token embedding $e(y \mid P)$ is defined as:

$$\mathrm{e}(y|P) = \begin{cases} \mathrm{M}_{op}(y) & \text{if } y \in V_{op} \\ \mathrm{M}_{con}(y) & \text{if } y \in V_{con} \\ \mathrm{h}^{p}_{loc(y,P)} & \text{if } y \in n_P \end{cases} \qquad \begin{matrix} +, -, \times, \div \\ \pi, 1, 2, 3, \ldots \end{matrix}$$

- Use **GRU** as encoder of context representation.
- Use **RNN** to encode a subtree in a bottom-up manner, which will play a role in the right sub-goal generation

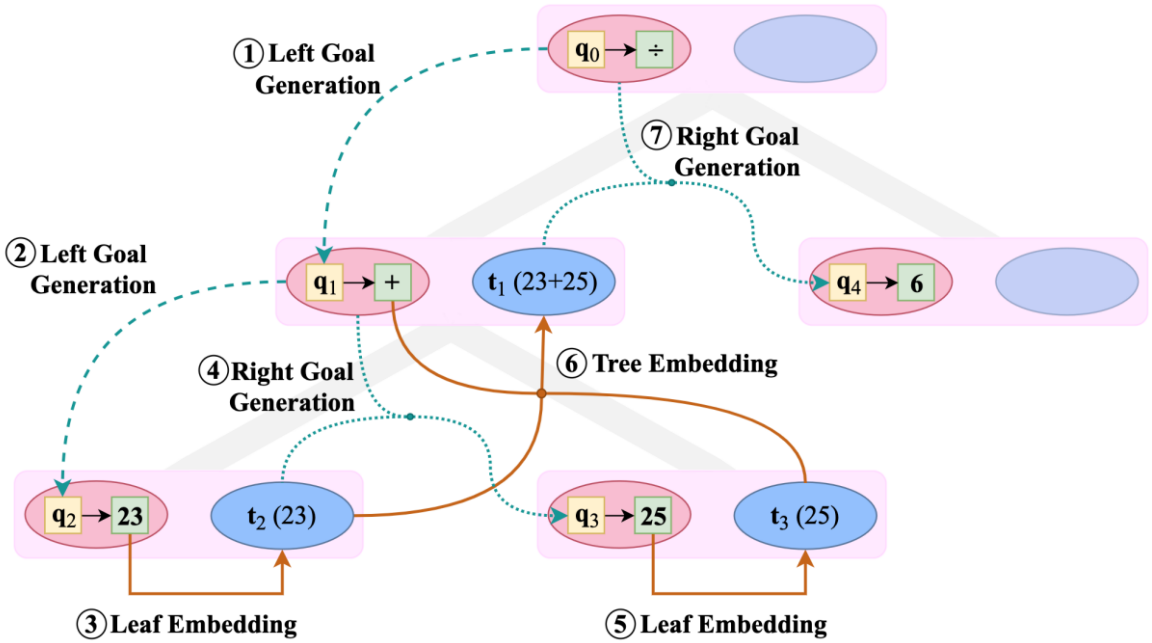
Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. IJCAI. 2019.

Part 1.

Background

- **Math Word Problem Solver: Seq2Tree**
  - Motivated by the commonly-used goal-driven mechanism in human problem solving
  - ➡ Generate expression trees by explicitly modeling tree-structured relationship between quantities

Goal-driven Tree-structured Model (GTS)



Model Architecture

Encoder

Root Goal Initialization and Token Embedding

Top-down Goal Decomposition

Left Sub-Goal Generation

Right Sub-Goal Generation

Subtree Emebedding via RNN

The output probability

$$p(T|P) = \prod_{t=1}^{m} \text{prob}(y_t|\mathbf{q}_t, \mathbf{c}_t, P)$$

Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. IJCAI. 2019.

Background

• Math Word Problem Solver: Seq2Tree

Root Goal Initialization and Token Embedding

$$\mathbf{h}_s^p = \overrightarrow{\mathbf{h}_s^p} + \overleftarrow{\mathbf{h}_s^p} \qquad \mathbf{q}_0 = \overrightarrow{\mathbf{h}_n^p} + \overleftarrow{\mathbf{h}_0^p}$$

Top-down Goal Decomposition

$$\text{score}(\mathbf{q}, \mathbf{h}_s^p) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{q}, \mathbf{h}_s^p]) \xrightarrow{\text{softmax}} a_s \Rightarrow \mathbf{c} = \sum_s a_s \mathbf{h}_s^p$$

$$\text{s}(y|\mathbf{q}, \mathbf{c}, P) = \mathbf{w}_n^\top \tanh\left(\mathbf{W}_s\left[\mathbf{q}, \mathbf{c}, \mathbf{e}(y|P)\right]\right)$$

$$\text{prob}(y|\mathbf{q}, \mathbf{c}, P) = \textbf{softmax}\ \text{s}(y|\mathbf{q}, \mathbf{c}, P)$$

$$\hat{y} = \arg\max \text{prob}(y|\mathbf{q}, \mathbf{c}, P)$$

Left Sub-Goal Generation

$$o_l = \sigma\left(\mathbf{W}_{ol}\left[\mathbf{q}, \mathbf{c}, \mathbf{e}(\hat{y}|P)\right]\right) \qquad g_l = \sigma\left(\mathbf{W}_{gl}\mathbf{h}_l\right)$$
$$C_l = \tanh\left(\mathbf{W}_{cl}\left[\mathbf{q}, \mathbf{c}, \mathbf{e}(\hat{y}|P)\right]\right) \qquad Q_{le} = \tanh\left(\mathbf{W}_{le}\mathbf{h}_l\right)$$
$$\mathbf{h}_l = o_l \odot C_l \qquad \mathbf{q}_l = g_l \odot Q_{le}$$

Right Sub-Goal Generation

$$o_r = \sigma\left(\mathbf{W}_{or}\left[\mathbf{q}, \mathbf{c}, \mathbf{e}(\hat{y}|P)\right]\right) \qquad g_r = \sigma\left(\mathbf{W}_{gr}\left[\mathbf{h}_r, \mathbf{t}_l\right]\right)$$
$$C_r = \tanh\left(\mathbf{W}_{cr}\left[\mathbf{q}, \mathbf{c}, \mathbf{e}(\hat{y}|P)\right]\right) \qquad Q_{re} = \tanh\left(\mathbf{W}_{re}\left[\mathbf{h}_r, \mathbf{t}_l\right]\right)$$
$$\mathbf{h}_r = o_r \odot C_r \qquad \mathbf{q}_r = g_r \odot Q_{re}$$

Subtree Emebedding via RNN

$$\mathbf{t} = \begin{cases} comb(\mathbf{t}_l, \mathbf{t}_r, \hat{y}) & \text{if } \hat{y} \in V_{op} \\ \mathbf{e}(\hat{y}|P) & \text{if } \hat{y} \in n_P \cup V_{con} \end{cases}$$

$$comb(\mathbf{t}_l, \mathbf{t}_r, \hat{y}) = g_t \odot C_t$$
$$g_t = \sigma\left(\mathbf{W}_{gt}\left[\mathbf{t}_l, \mathbf{t}_r, \mathbf{e}(\hat{y}|P)\right]\right)$$
$$C_t = \tanh\left(\mathbf{W}_{ct}\left[\mathbf{t}_l, \mathbf{t}_r, \mathbf{e}(\hat{y}|P)\right]\right)$$

Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. IJCAI. 2019.

Part 1.

Background

- **Math Word Problem Solver: Seq2Tree**
  - Case 1: Avoid generating mathematically invalid expressions
    - Sequence of pre-order traversal can be guaranteed to be computable
  - Case 2: Avoid predicting spurious numbers (e.g., "$n4$")
    - "$n4$" can not be converted back to a number
    - The effective size of target vocabulary is set dynamically according to the specific problem

★ Token embedding $e(y \mid P)$ involves n_p V_{op} V_{Con}

Typical cases. Note that the results are represented as pre-order traversal of expression trees

Case 1: The store shipped in a batch of leather shoes. NUM(n_0 [$\frac{1}{3}$]) of the total was sold on the first day, and NUM(n_1 [$\frac{3}{5}$]) of the first day's sale was sold on the second day. There were NUM(n_2 [280]) pairs left. How many pairs of leather shoes did the store bring in?

Seq2Seq: $\div n_2 - 1 n_0 * n_0 n_1$;**(error)** **GTS:** $\div n_2 - - 1 n_0 * n_0 n_1$;**(correct)**

Case 2: Of the NUM(n_0 [697]) combined shipment equipments of Shenzhou NUM(n_1 [7]) spacecraft, NUM(n_2 [346]) are followed, NUM(n_3 [237]) are updated, and the rest are newly developed. How many new equipments are there?

Seq2Seq: $- - n_0 n_3 n_4$;**(error)** **GTS:** $- - n_0 n_2 n_3$;**(correct)**

Zhipeng Xie et al. A Goal-Driven Tree-Structured Neural Model for MathWord Problems. IJCAI. 2019.

Part 1.

Background

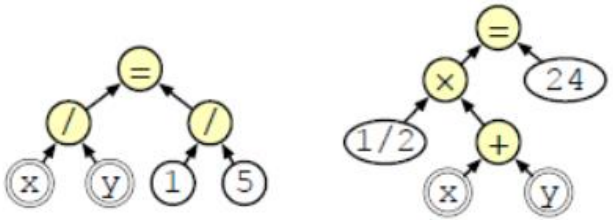
- **Math Word Problem Solver:** **Seq2Tree ➡ Seq2Exp**
  - Seq2Tree methods focus on arithmetic problems which only have one variable
    - The result of arithmetic problem are simpler that have only one equation or expression
  - Cannot solve equation set problems that have multiple equations and variables
    - Extending them to decode multiple trees for equation set problems would require substantial revisions
  - For example, as they have to decode multiple equations one by one, the order of equations have to be determined
    - Equations in an equation set are unordered intrinsically
    - Determine their order puts unnecessary burden on models

➡ **Humans compose sub-expressions when reading and understanding the problem**

Input
Problem: The ages of Tom and his father are in the ratio of 1: 5, 1/2 of their sum is 24. Find their ages.

Output (In different forms)
Answer: 8, 40

Equations: $x/y = 1/5, \quad \frac{1}{2} \times (x + y) = 24$
Equation Trees:



Yixuan Cao et al. A bottom-up dag structure extraction model for math word problems. AAAI. 2021.

⬅ **Seg2DAG is first SeqEXP Model**

Part 1.

Background

- **Today here to introduce MWP-NAS:**
  Non-autoregressive MWP solver (Seq2Exp)
  - A goal-driven manner for multi-branch decomposition to generate the MTree of expressions
  - A cross-goal attention strategy to pass information between goals during goal decomposing
  - Design two metrics based on MTree for better expression evaluation

Motivation			
A goal-driven manner	←	Seq2Tree	Goal-driven Tree-structured Model (GTS) (Xie and Sun, 2019)
Multi-branch decomposition	←	Seq2Exp	Structure-Unified M-Tree Coding Solver (SUMC-Solver) (Bin Wang et al., 2022)
Cross-goal Aattention	←	Prev Work	Non-autoregressive Transformer (Gu et al., 2018)
			Pointer Network (Peter Brown et al., 2018)

Previous Work

Structure-Unified M-Tree Coding Solver for Math Word Problem

ACL 2022

Bin Wang, Jiangzhou Ju, Yang Fan, Xinyu Dai* , Shujian Huang, Jiajun Chen

National Key Laboratory for Novel Software Technology, Nanjing University

Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing

Background

- **Seq2Exp: When a problem has multiple correct answer**

Multiple correct answer for problem		
$2 \times 3 + 4 + 5$	$3 \times 2 + (5 + 4)$	$5 + 3 \times 2 + 4$

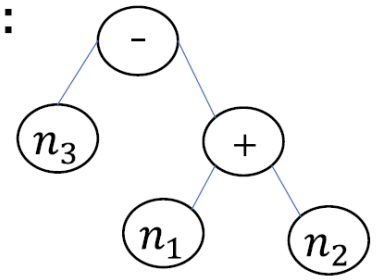
◦ A large non-deterministic output space
  - The number of different expression sequences or binary trees can grow large with combinations
  - The knowledge learned by the model will be incomplete with only one answer obtained by the solver

◦ Data-driven method
  - The demand for data will also increase
  - Most data-driven methods perform poorly under low-resource conditions

Part 1.

Background

- **Template-Based Math Word Problem Solvers with RNN (Wang et al, 2019)**
  - Use structural template with suffix expression, to annotate the math problems
    - Apply a seq2seq model to predict a tree-structure template
  - Equation normalization to further reduce the number of possible templates
    - Template prediction: Fill the unknown operators with the derived template
  - RNN to infer the inner nodes

Step 1	**Original Expression :** n_3 **-** $(n_2 +$ $n_1)$
Step 2	**Re-ordered Expression :** n_3 **-** $(n_1 +$ $n_2)$
Step 3	**Expression Tree :** 
Step 4	**Postfix Expression :** n_3 n_1 n_2 + -

		MAWPS	Math23K
Classification	Bi-LSTM	62.8	57.9
	Self-Att	60.4	56.8
Our Approach	T-RNN	**66.8**	**66.9**
	- EN	63.9	61.1
	- Bi-LSTM	31.1	34.1
	- Self-Att	66.3	65.1

Table 3: Accuracy of template prediction module.

	MAWPS	Math23K
Accuracy w/o EN	62.2	59.6
Accuracy w EN	65.8	69.1
Percentage of illegal templates	9.0	0.7

Lei Wang et al. Template-based math word problem solvers with recursive neural networks. AAAI. 2019.

Background

- **Output diversity increases the difficulty of model learning**
  - We analyze the causes for the output diversity

- **The causes for the output diversity**
  - Uncertainty of computation order of the mathematical operations:
    - Giving the same priority to the same or different mathematical operations

$$n_1 + n_2 + n_3 - n_4$$

Brackets can also lead to many equivalent outputs with different forms (binary trees)

$n_1 + n_2 - n_3$	$n_1 - (n_3 - n_2)$	$(n_1 + n_2) - n_3$

  - The uncertainty caused by the exchange of operands or sub-expressions
    - Addition & multiplication have the property that the operands or sub-expressions of both sides are allowed to be swapped

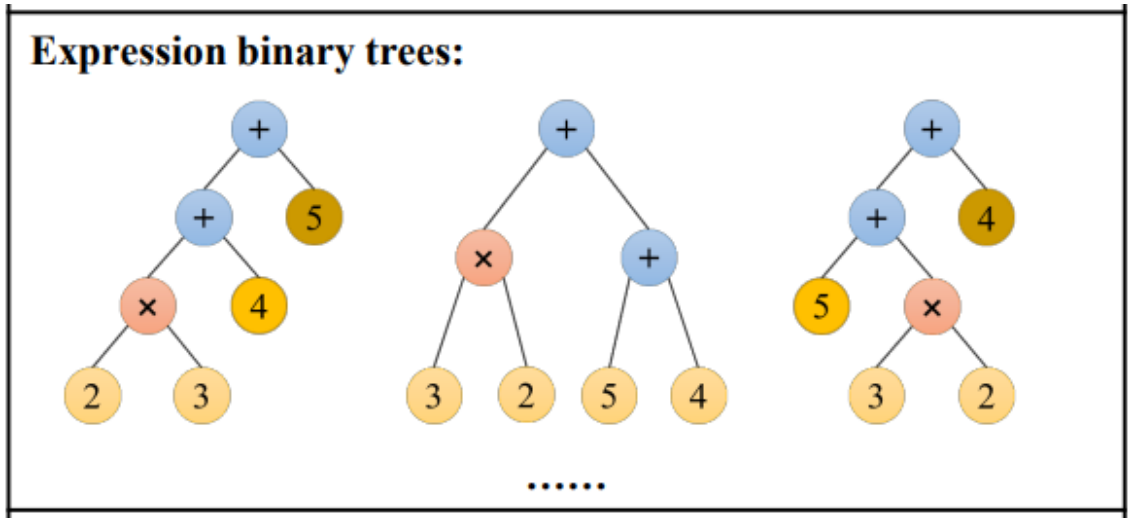
Introduction

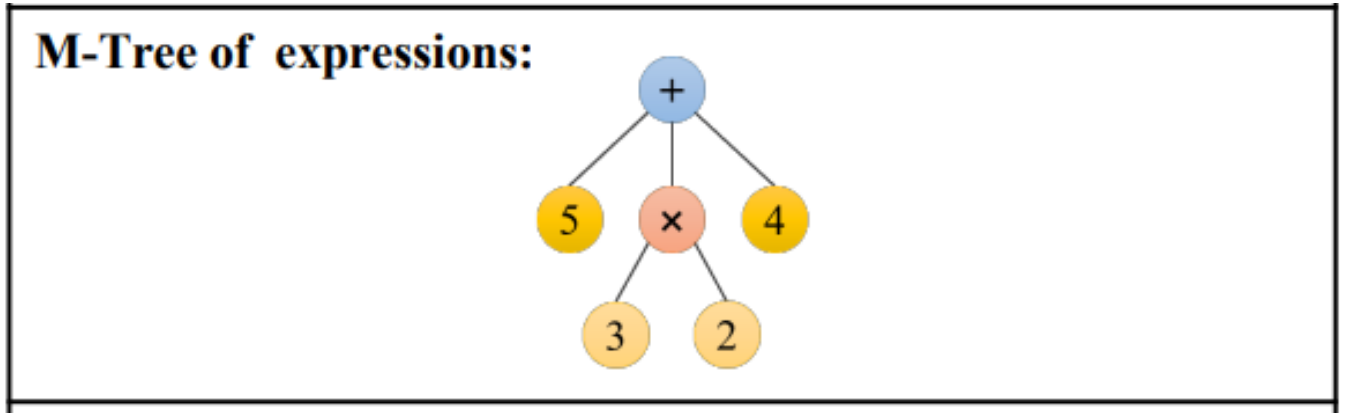
- **Structure-Unified M-Tree Coding Solver (SUMC-Solver)**
  - Existing work (Xie and Sun, 2019; Wu et al., 2020, 2021b)
    - Taking advantage of the tree structure information of MWP expressions can achieve better performance
  - Retain the use of a tree structure but further develop on top of the binary tree with an M-tree which contains any M branches
  - The ability of the M-tree to unify output structures is reflected in both horizontal and vertical directions

Part 2.

Introduction

- **The uncertainty of computation orders for operations**
  - Set the root to a specific operation and allow any number of branches for internal nodes in the M-tree
  - Reduce the diversity of the tree structure in the **vertical direction**

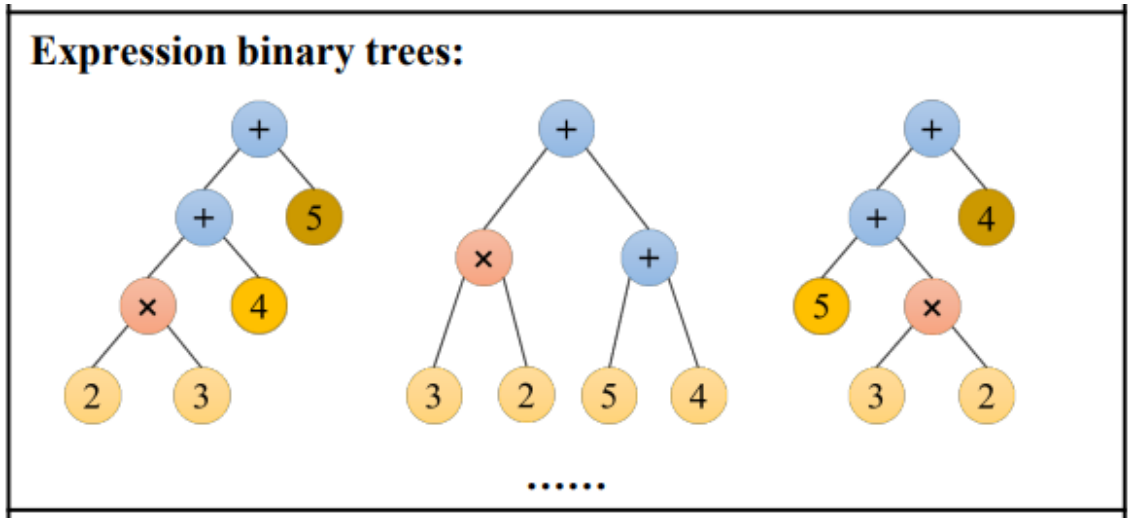


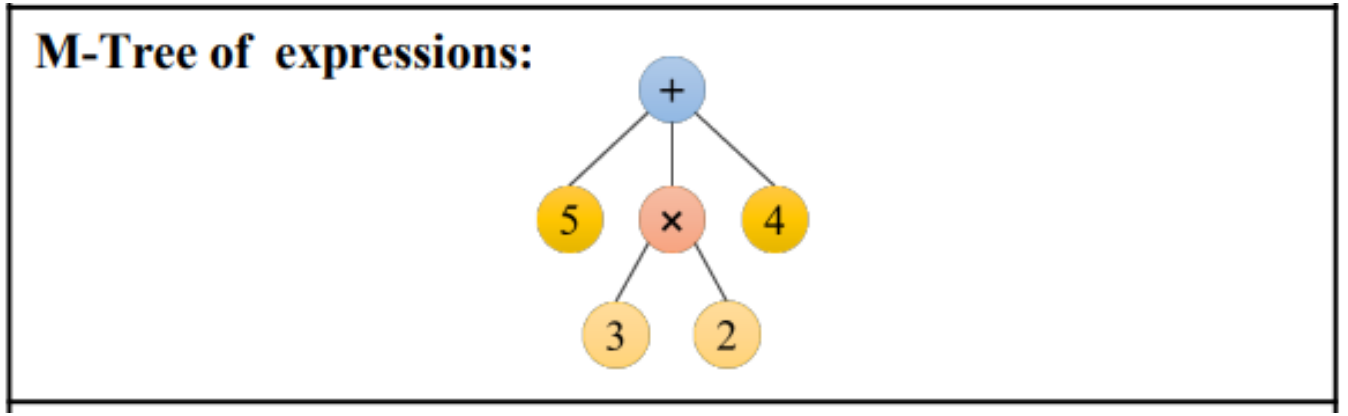


Part 2.

Introduction

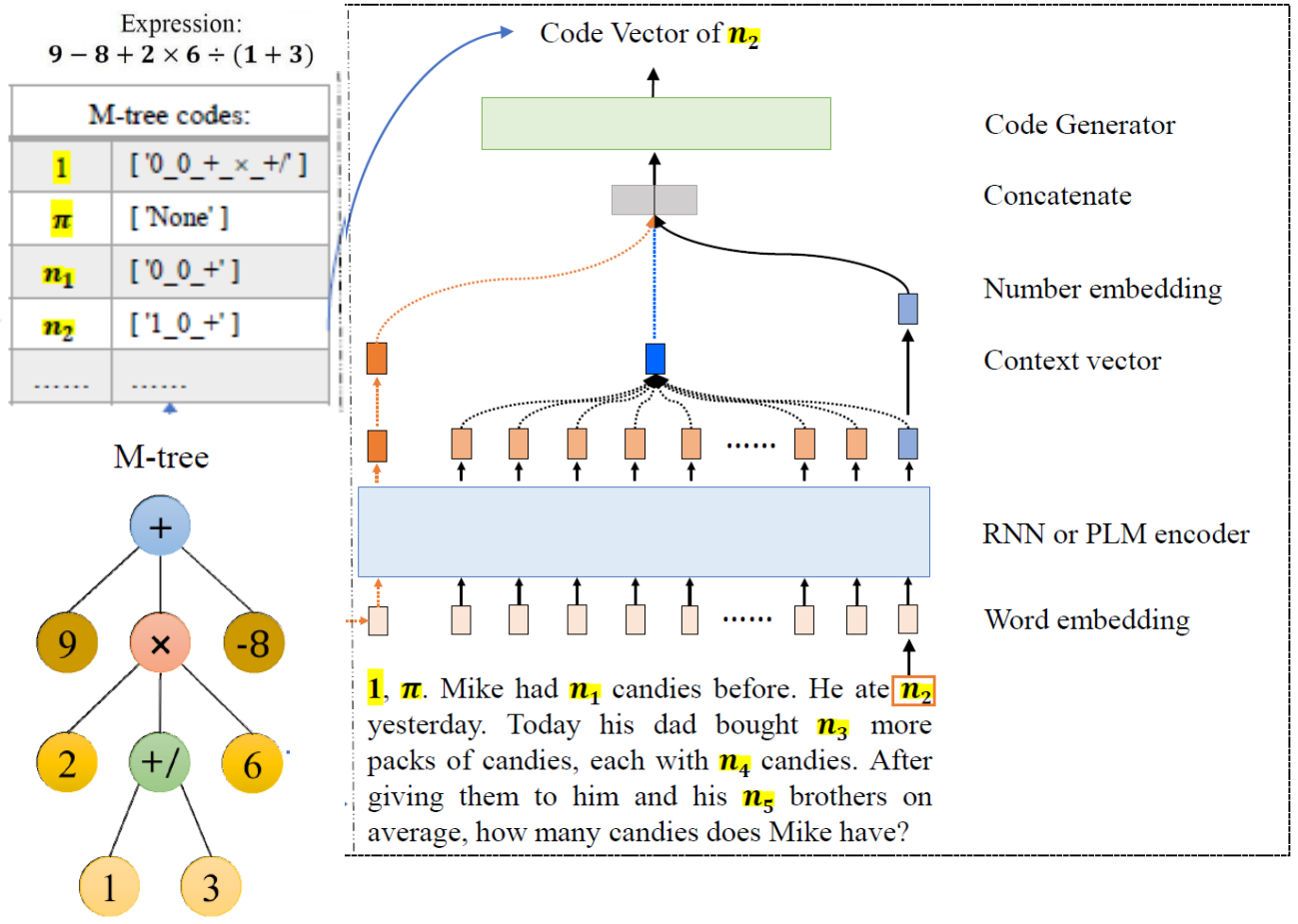
- **The uncertainty caused by the exchange between the left and right sibling nodes in original binary trees**
  - Redefine the operations in the M-tree to make sure that the exchange between any sibling nodes will not affect the calculation process
  - Treat M-trees that differ only in the left-to-right order of their sibling nodes as the same
  - With this method, the structural diversity in the **horizontal direction** is also reduced





Introduction

- **M-tree codes & a seq2code framework for M-tree learning**

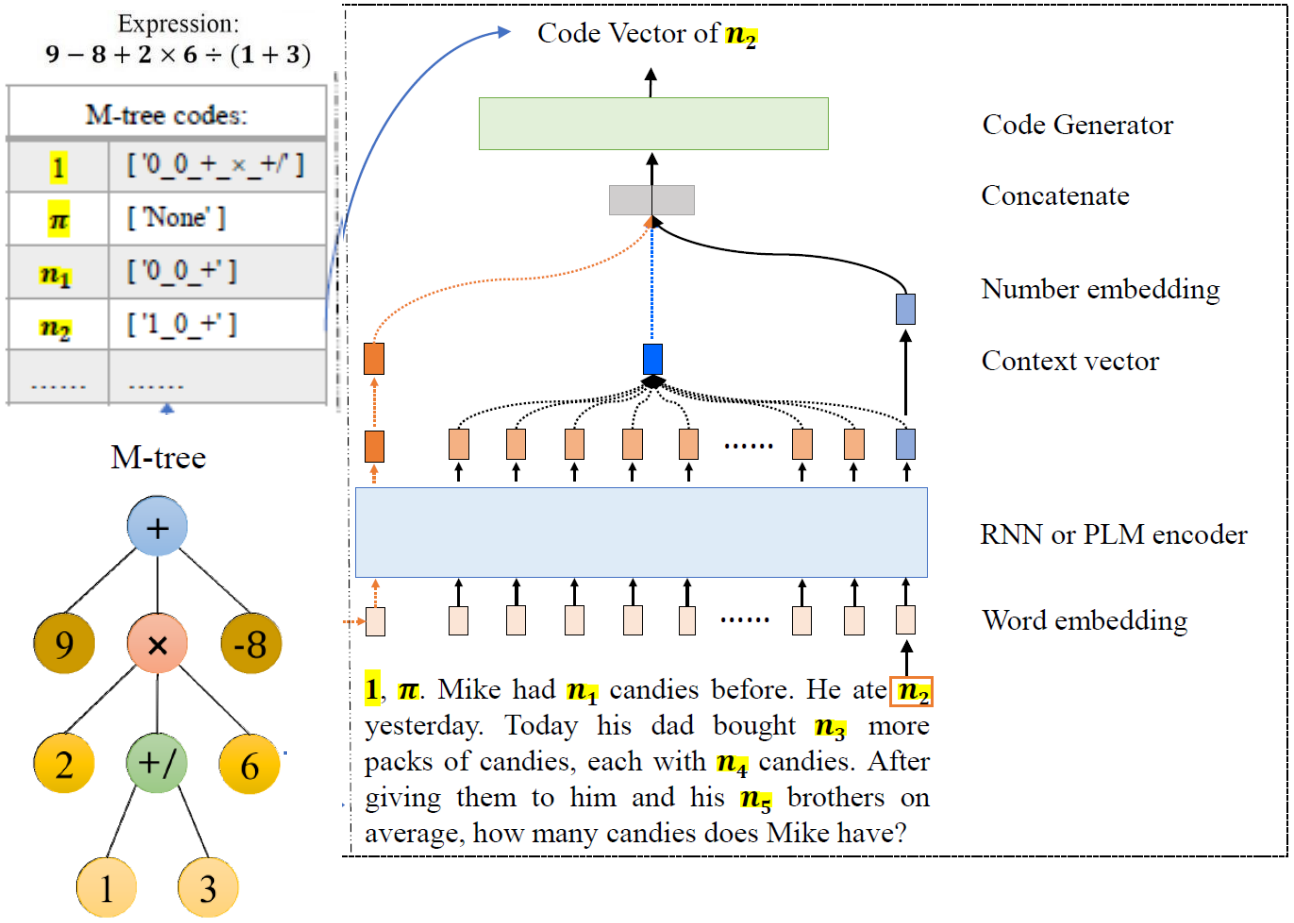


M-tree codes

- Abandon the top-down & left-to-right autoregressive generation used for binary trees
  - Because can not avoid the diversity caused by the generation order of sibling nodes
- Instead, Encode the M-tree into M-tree codes that can be restored to original M-tree
  - The codes store 1) the information of the paths from the root to leaf nodes & leaf nodes themselves

Introduction

- **M-tree codes & a seq2code framework for M-tree learning**



A seq2code framework

- Inspiration by the sequence labeling methods
- Generate the M-tree codes in a non-autoregressive way:
  - Takes the problem text as the input sequence
  - Outputs the M-tree codes of the numbers (numerical values) in the math word problem
- Then restore the codes to a M-tree
  - Represent the calculation logic between the numbers
  - Finally calculate the answer

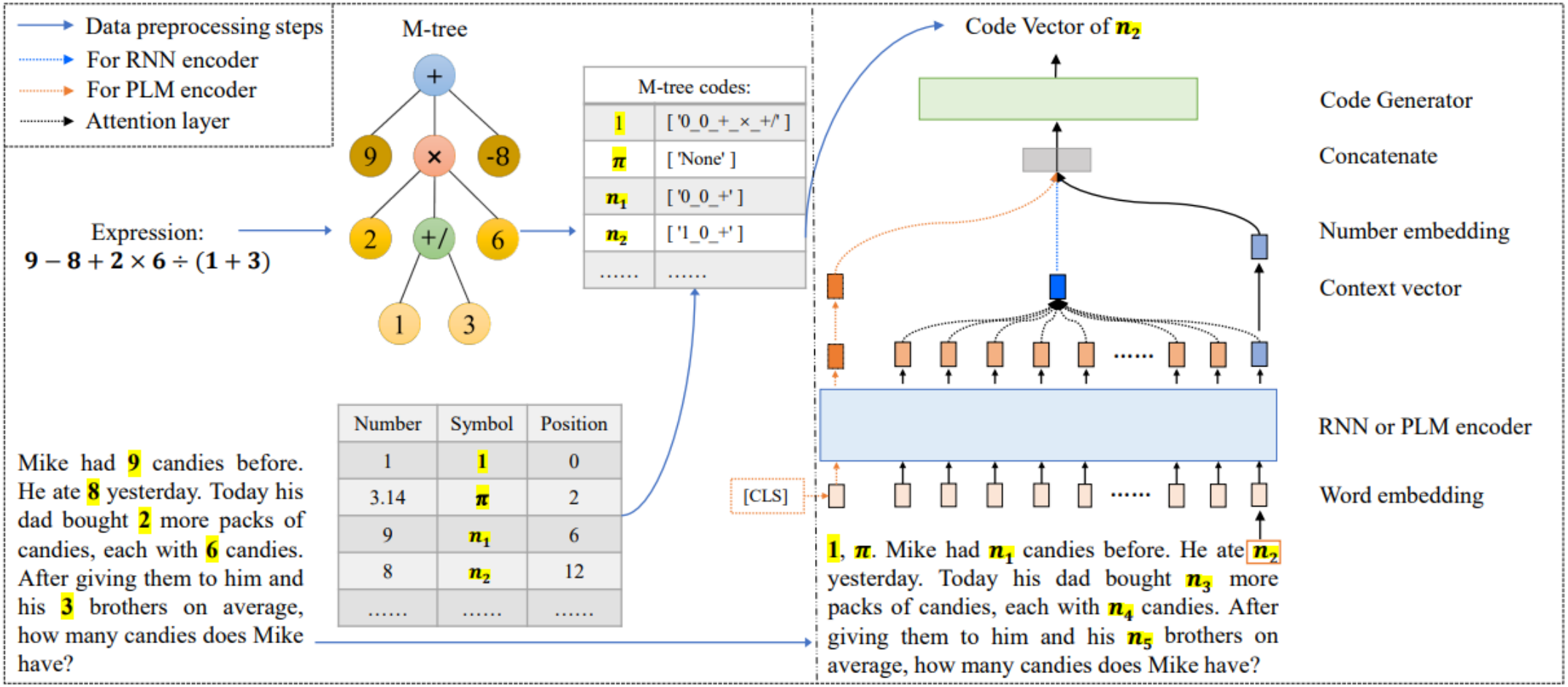
Part 3.

Method

• The Design of SUMC-Solver

MWP with the M-tree and M-tree codes

Main architecture of our seq2code model



Method

- **Problem Definition**
  - A math word problem
    - A sequence of tokens, where each token can be either a word or a numerical value
  - Input sequence
    - constants, including 1 and π, are required

$$X = (x_1, x_2, ..., x_n)$$

  - All the numerical values that appear in X

$$V = \{v_1, v_2, ..., v_m\}$$

  - $\mathbf{c}_i$ is a target code vector for v_i

$$C = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_m\}$$

Method

- **M-Tree: Data Pre-processing**
  - Add additional constants (e.g., 1 and π) that may be used to the front of the input sequence
  - Replace each numerical value v_i with a special symbol
  - Prepare for the conversion of expression to the M-tree
    - Remove all the brackets of the expression by using the SymPy2 Python package

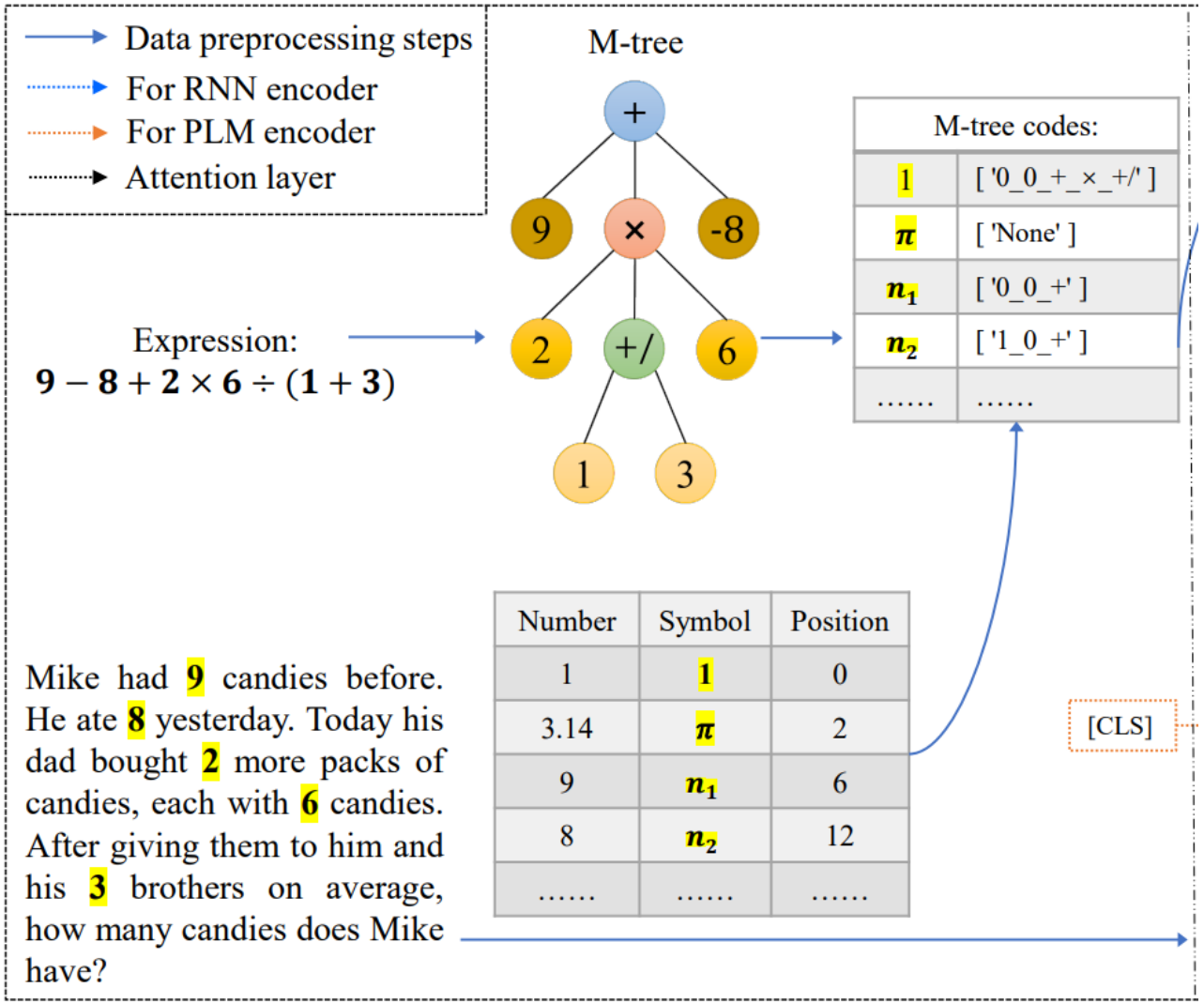
$$n_1 \times (n_2 \pm n_3) \longrightarrow n_1 \times n_2 \pm n_1 \times n_3$$

$$n_1 + (n_2 \pm n_3) \longrightarrow n_1 + n_2 \pm n_3$$

$$a^b \longrightarrow n_1 * n_1 * \ldots * n_1$$

Method

• M-Tree: The Design



M-tree codes

◦ Internal node

- Each internal node has any M branches, where M is an integer greater than or equal to 1
- Four types of internal nodes, corresponding to redefined operations $\{+, \times, \times-, +/\}$

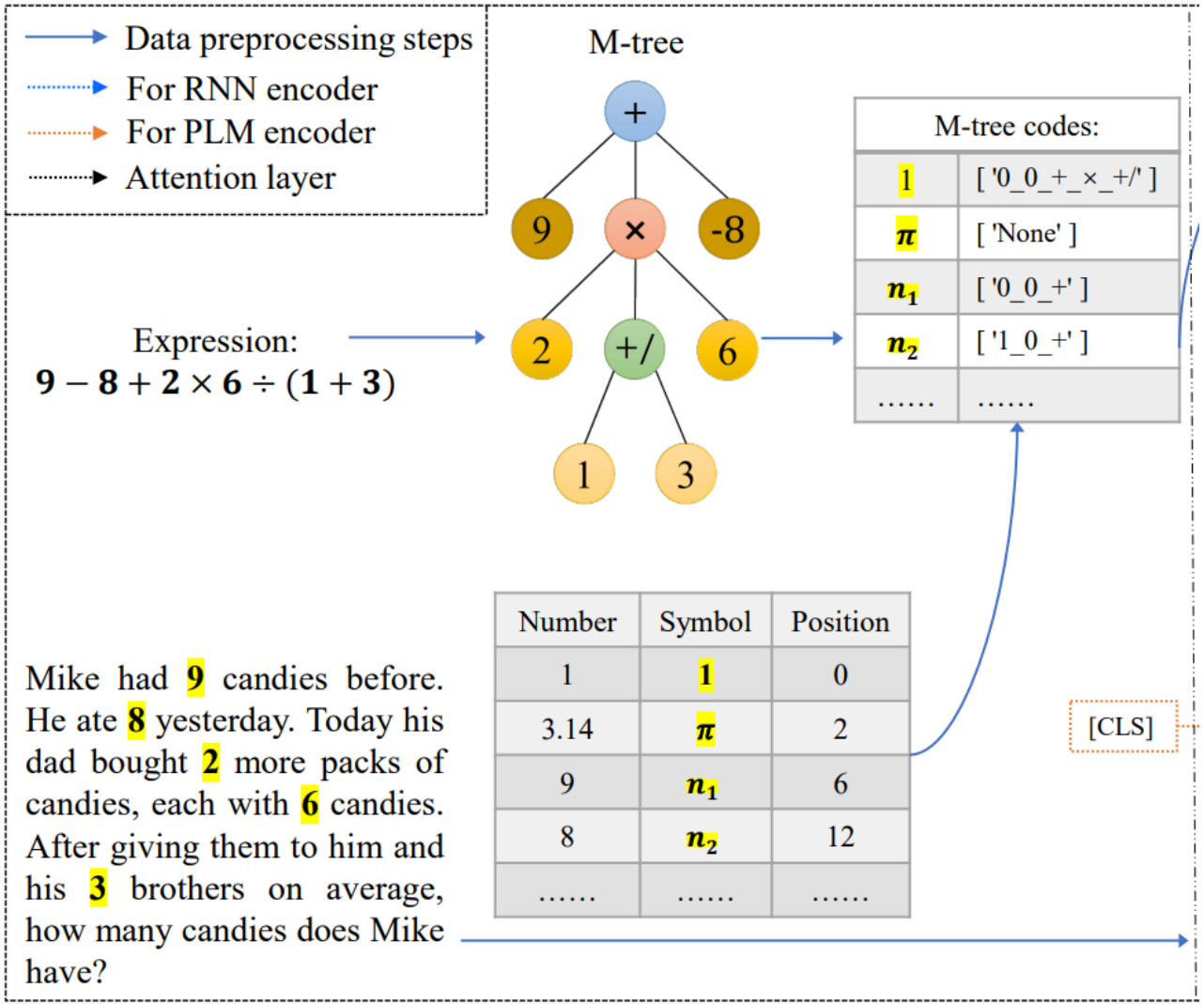
◦ Leaf node

- Four types of leaf nodes, corresponding to the numerical value: $\{v, -v, \frac{1}{v}, -\frac{1}{v}\}$

Part 3.

Method

- **The implementation details of the M-tree**



$\{+, \times, \times-, +/\}$

- Ensure sibling nodes are structurally equivalent in the M-tree
- Ensure two M-trees that differ only in the order of their sibling nodes will be treated as the same

- For an internal node that has k children $\{v_1, v_2, ..., v_k\}$
- The node of "+" ("×") :

$$v_1 + v_2 +, ..., +v_k \ (v_1 \times v_2 \times, ..., \times v_k)$$

- The node of "×−"("+/") :

$$-v_1 \times v_2 \times, ..., \times v_k \ (\frac{1}{v_1 + v_2 +, ..., +v_k})$$

Method

- **The implementation details of the M-tree**
  - Follow the order of priority for operations $> (\times = \div) > (+ = -)$
  - Convert the operations one-by-one in the expression $\{+, \times, \times-, +/\}$

$$v_1 \div v_2 \Longrightarrow v_1 \times v_2' \qquad v_1 - v_2 \Longrightarrow v_1 + v_2'$$

$$v_1 - v_2 \times v_3 \Longrightarrow v_1 + v_2(\times-)v_3 \qquad v_1 \div (v_2 + v_3) \Longrightarrow v_1 \times v_2(+/)v_3$$

  - After obtaining the new expression,
    Convert it to a binary tree and then reduce it from top to bottom to get the final M-tree
    - The parent node v_p The child node v_c

$$\text{“}v_p = \times \text{ and } v_c = \times-\text{”} \Longrightarrow \text{“}v_p = \times-\text{”}$$

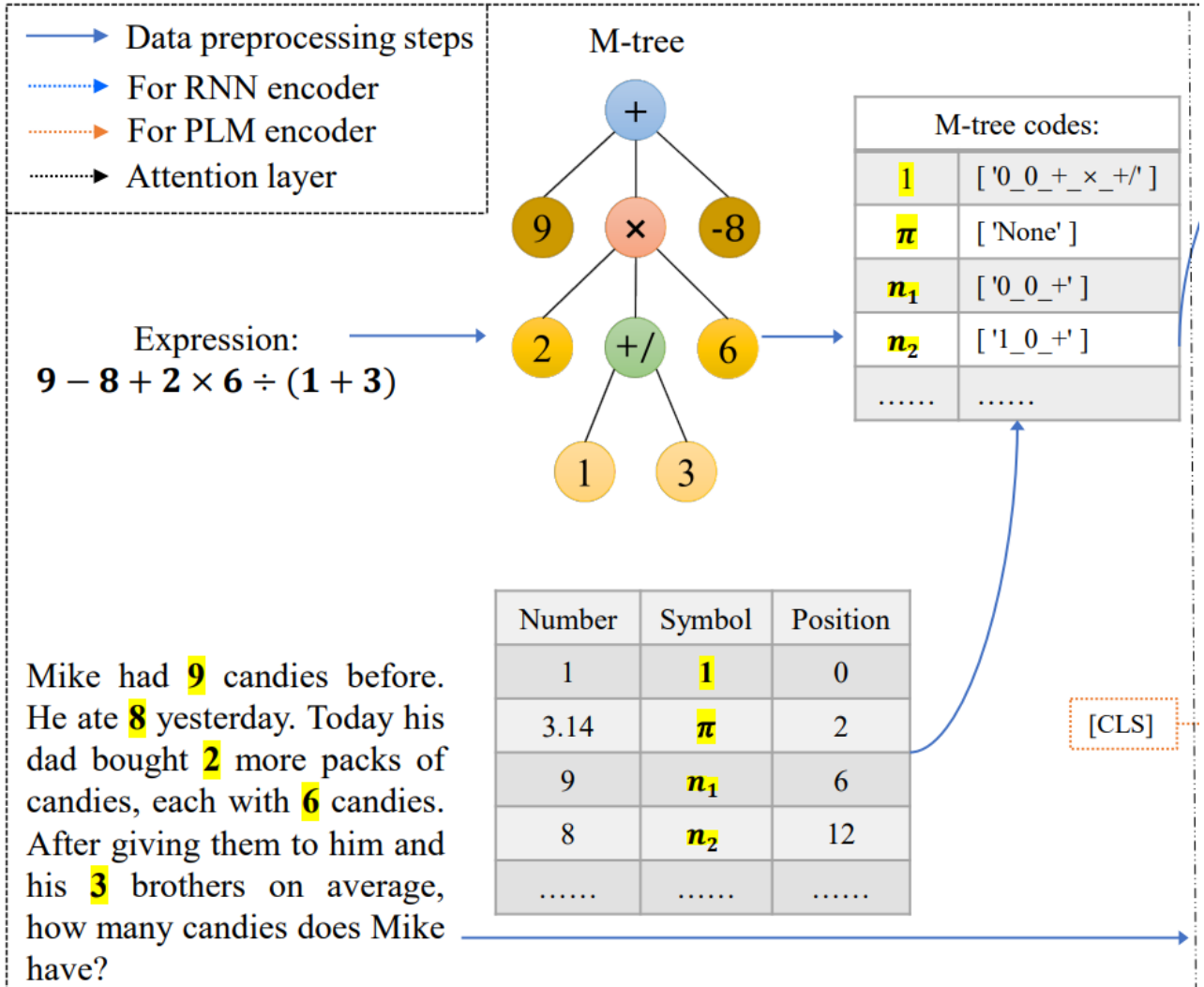
$$\text{“}v_p = \times- \text{ and } v_c = \times-\text{”} \Longrightarrow \text{“}v_p = \times\text{”}$$

Method

- **The Design of M-Tree Codes**
  - M-Tree: Autoregressive-based generation cannot avoid the diversity caused by the sequential order of sibling nodes at the output side
    - Since the nodes in the M-tree can have any number of branches and sibling nodes are structurally equivalent
  - M-Tree Codes: Encode the structure information of the M-tree into each leaf node
    - Form a mapping between the M-tree and the codes set of leaf nodes
    - The model can generate the codes in a non-autoregressive way

Method

• Components of M-tree Codes



Each leaf node

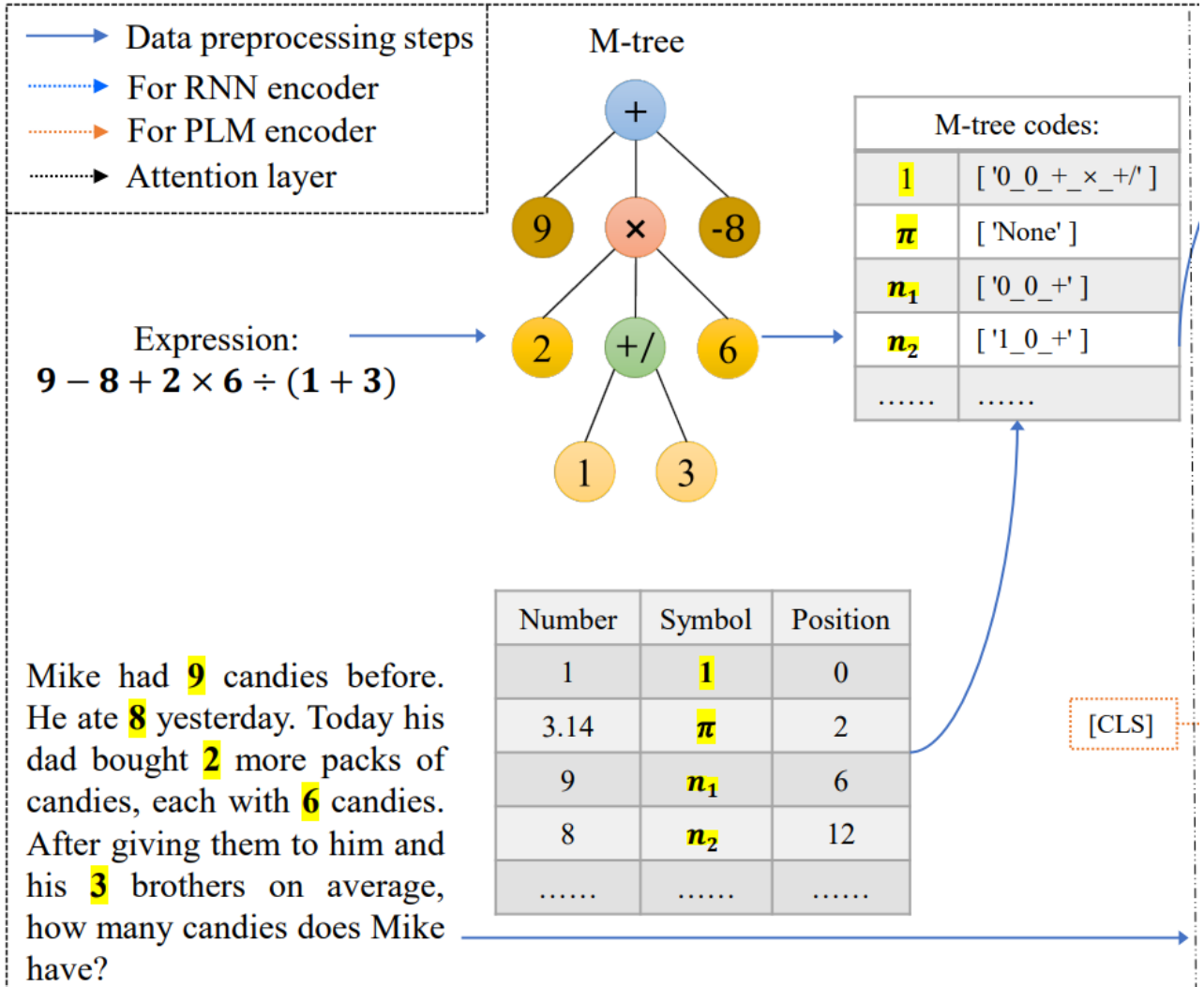
- The numerical value
  - If $v_i' = v_i$, the code is set as "0_0";
  - If $v_i' = -v_i$, the code is set as "1_0";
  - If $v_i' = \frac{1}{v_i}$, the code is set as "0_1";
  - If $v_i' = -\frac{1}{v_i}$, the code is set as "1_1";
- The sequential operation symbols of all internal nodes on the path from the root to the current leaf node v_i'
- If the internal nodes that are siblings have the same type (e.g., all " × " nodes)
  - Need to be marked with a special symbol added to the end to distinguish them from each other

Method

- **Vector Representation of M-tree Codes**
  - The final code vector c_i for model learning will be obtained based on The final set of M-tree codes
  - Considering that the value v_i that appears only once in the input problem text may appear multiple times in the M-tree
    - For example, in "$v_i \times v_j \pm v_i \times v_k$" , v_i will appear in two leaf nodes and have two identical or different M-tree codes
  - Consequently, the set of numerical values $V = \{v_1, v_2, ..., v_m\}$ is mapping to a set of i-dimensional non-one-hot vectors:
  - $\mathbf{C} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_m}\}$, the value of ci in the k-th dimension indicates how many codes of that vi has

Method

• Vector Representation of M-tree Codes



The final code vector

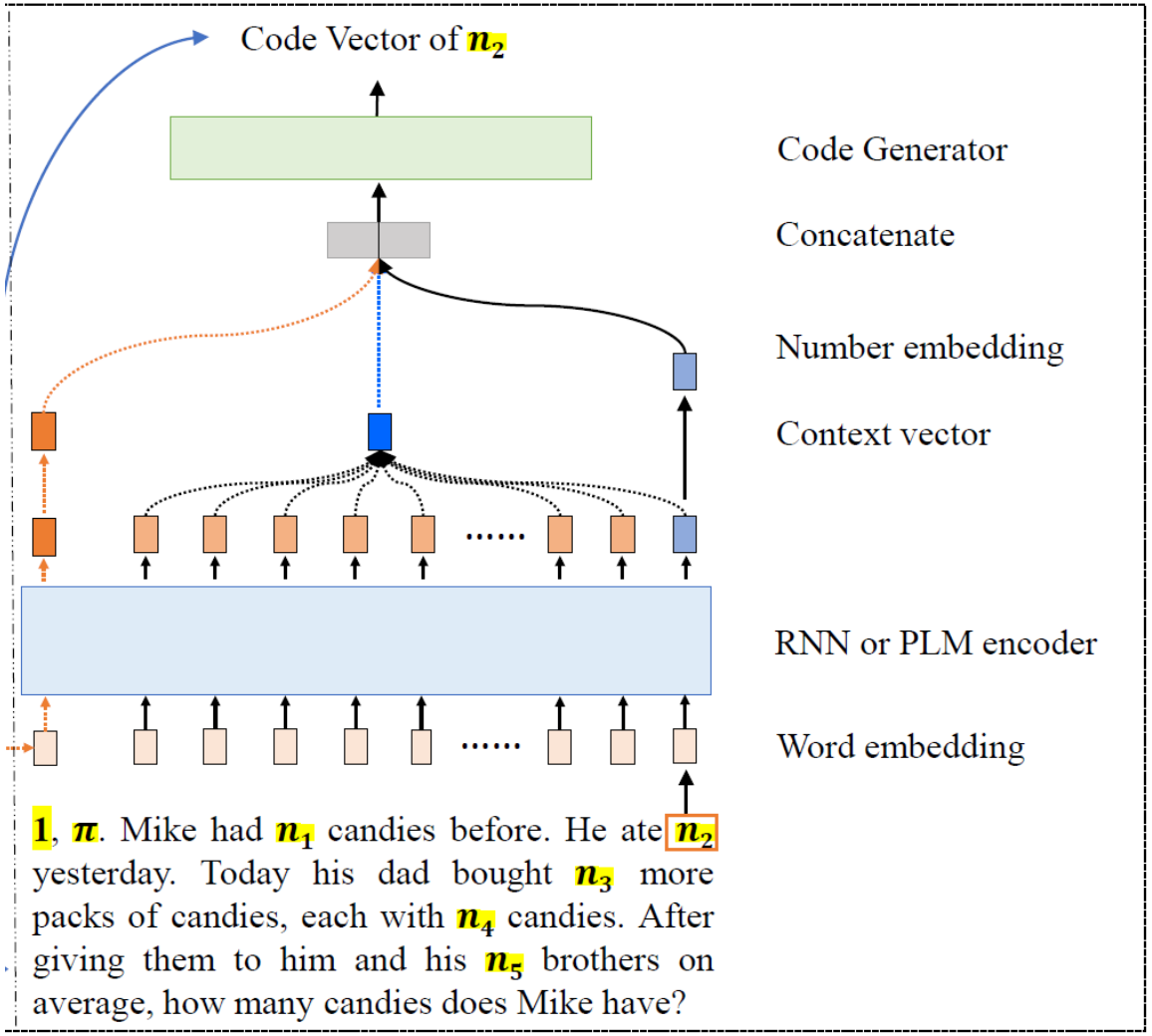
- Value "π" will be set as $[1, 0, \ldots, 0]^\top$

$[1, 0, \ldots, 0]^\top$

- Only the first dimension has the value of 1 indicating that "π" has only one M-tree code
- "None" means that it does not appear in the M-tree

Method

• Sequence-to-Code Model: RNN encoder (Problem Encoder)



- An encoder to transform the words of a MWP into vector representations
- Encodes the input sequence into a sequence of hidden states $\mathbf{h}_t^x = \left[\overrightarrow{\mathbf{h}_t^x}, \overleftarrow{\mathbf{h}_t^x}\right]$

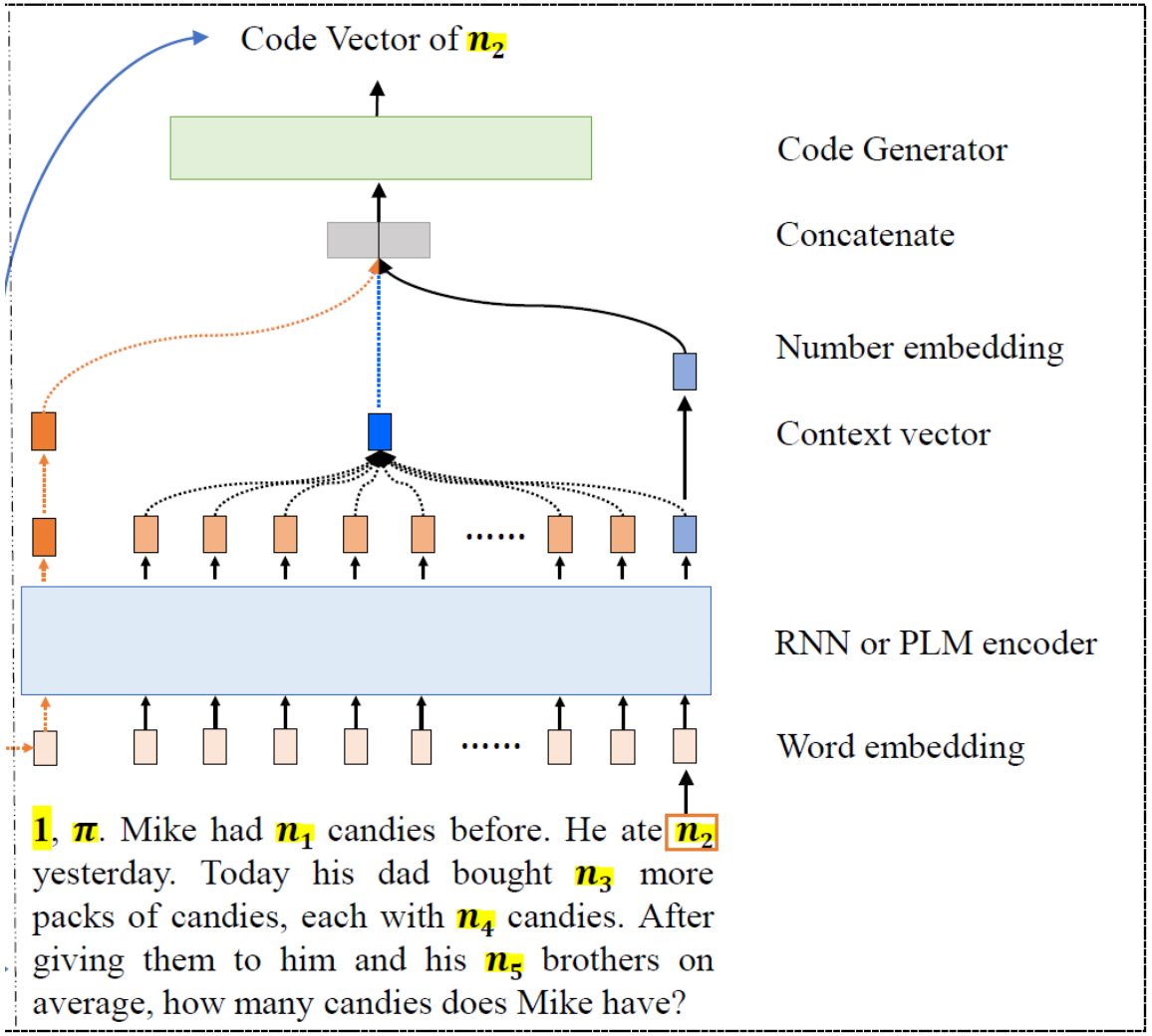
$$\overrightarrow{\mathbf{h}_t^x}, \overrightarrow{\mathbf{c}_t^x} = BiLSTM\left(\mathbf{e}_t^x, \overrightarrow{\mathbf{c}_{t-1}^x}, \overrightarrow{\mathbf{h}_{t-1}^x}\right)$$

$$\overleftarrow{\mathbf{h}_t^x}, \overleftarrow{\mathbf{c}_t^x} = BiLSTM\left(\mathbf{e}_t^x, \overleftarrow{\mathbf{c}_{t-1}^x}, \overleftarrow{\mathbf{h}_{t-1}^x}\right) \qquad \mathbf{e}_i^c = \mathbf{h}_{q_i}^x$$

- $\boldsymbol{e_t^x}$ the word embedding vector
- For the numerical value $\boldsymbol{v_i}$ in the problem
- Its semantic representation $\boldsymbol{e_i^c}$ is modeled by the corresponding BiLSTM output vector

Method

• Sequence-to-Code Model: RNN encoder (Problem Encoder)



- Better capture the relationship between different numerical values and the relationship between v_i and the unknown value to be solved (answer of the problem)

- Use an attention layer to derive a context vector E_i for v_i
- Context vector is expected to summarize the key information of the input problem
- The context vector E_i is calculated as a weighted representation of the source tokens:

$$\text{score}\left(\mathbf{e}_i^c, \mathbf{h}_t^x\right) = \mathbf{U}^\top \tanh\left(\mathbf{W}\left[\mathbf{e}_i^c, \mathbf{h}_t^x\right]\right) \Rightarrow \alpha_{it}$$

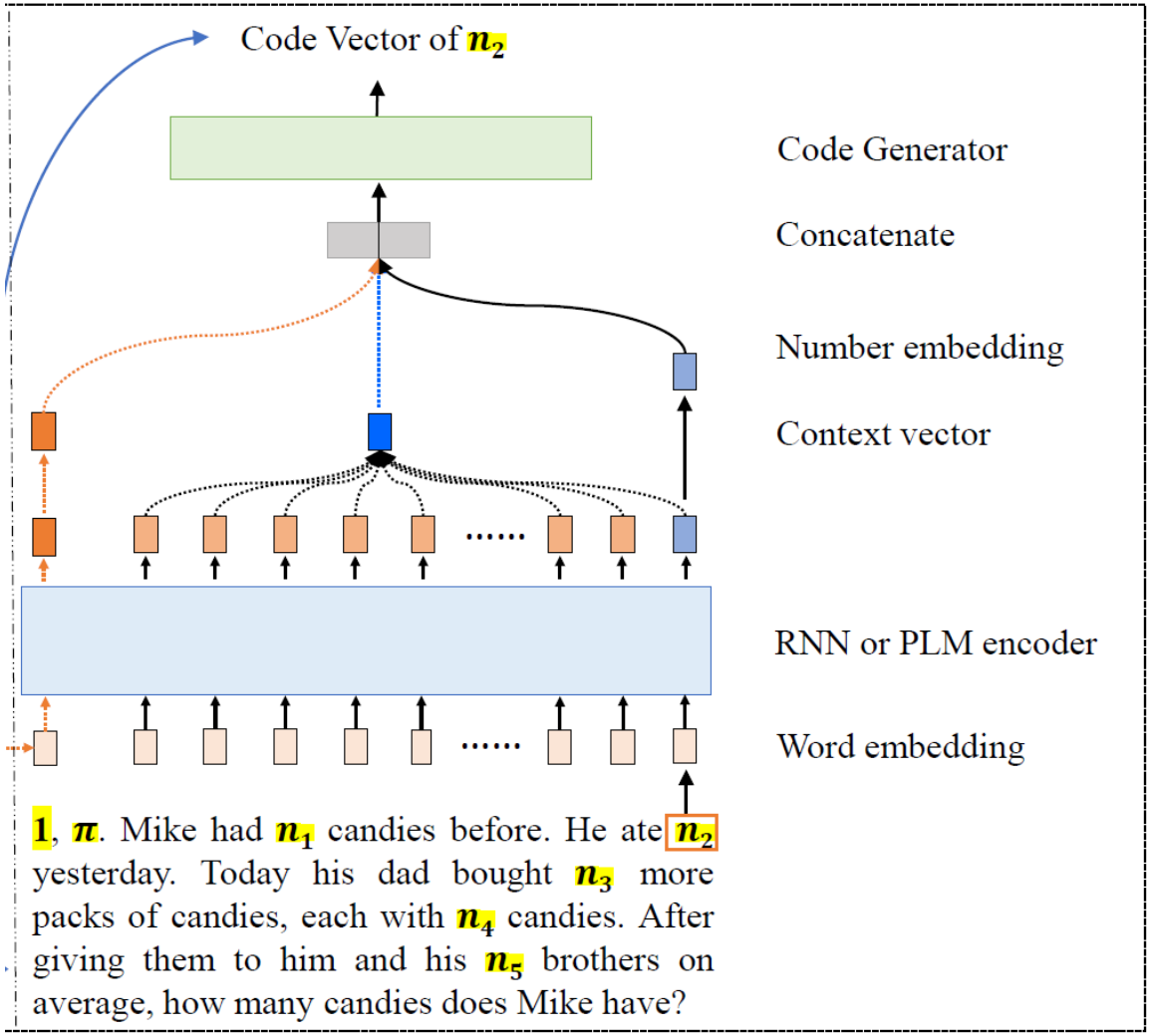
softmax

$$\mathbf{E}_i = \sum_t \alpha_{it} \mathbf{h}_t^x$$

- Obatin the input of the generator $\mathbf{z}_i^c = [\mathbf{E}_i, \mathbf{e}_i^c]$

Method

• Sequence-to-code Model: Code Generator



- Use a simple three-layer FFNN to implement the generator
- With the input $\boldsymbol{Z_i^c}$, the final code vector $\boldsymbol{c_i'}$ is generated
- σ an activation function

Code Generator

$$\mathbf{z}_{i1}^{c} = \sigma\left(\mathbf{z}_{i}^{c\top}\mathbf{W}_1 + \mathbf{B}_1\right) \qquad \mathbf{z}_{i2}^{c} = \sigma\left(\mathbf{z}_{i1}^{c\top}\mathbf{W}_2 + \mathbf{B}_2\right)$$

$$\mathbf{c}_i^{'} = \mathbf{z}_{i2}^{c\top}\mathbf{W}_3 + \mathbf{B}_3$$

Training Objective

$$\mathcal{L} = \sum_{(X^i,C^i)\in\mathbf{D}} \sum_{\mathbf{c}_i\in C^i} \mathcal{L}_{MSE}(\mathbf{c}_i, \mathbf{c}_i^{'}) \quad \mathcal{L}_{MSE}(\mathbf{c}_i, \mathbf{c}_i^{'}) = \frac{1}{l}\sum_{j=1}^{l}\left(\mathbf{c}_{ij} - \mathbf{c}_{ij}^{'}\right)^2$$

Experiment

- **Datasets**
  - MAWPS (Koncel-Kedziorski et al., 2016)
    - 2,373 problem
    - Performance with five-fold cross-validation, pre-processing method, to avoid coarsely filtering out too much data
  - Math23K
    - Public test set

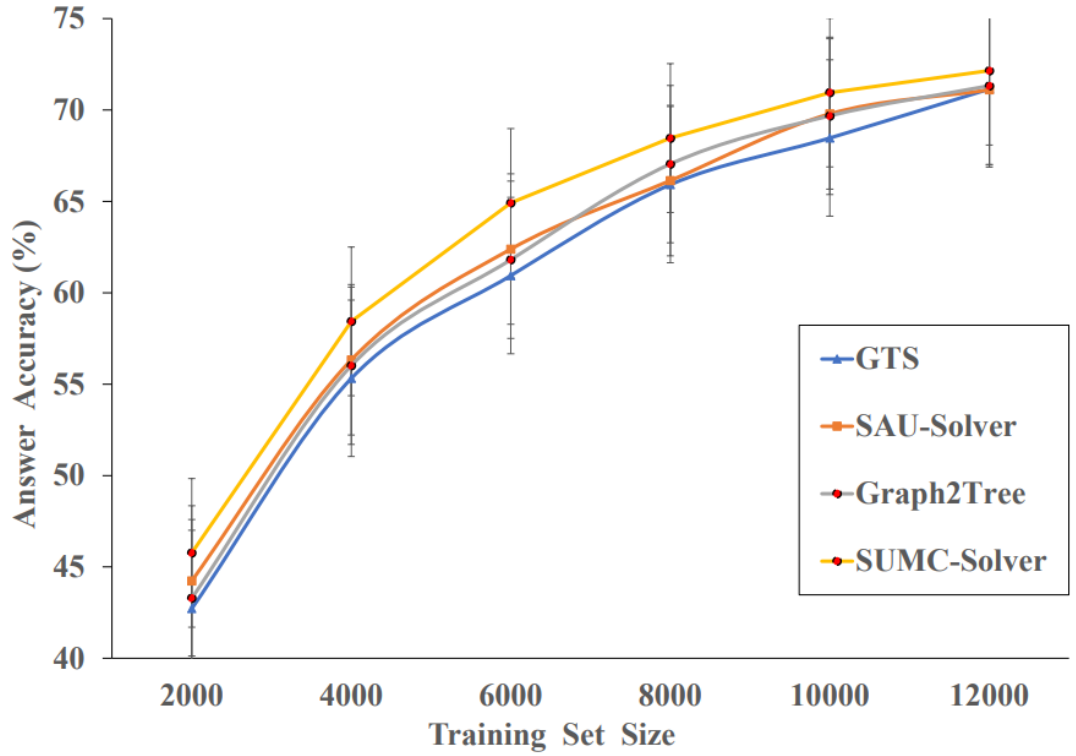
- **Evaluation Metric**
  - Answer accuracy
    - If the value predicted by the solver equals the true answer, it is thought of as correct

- **Baseline**
  - RNN encoder: word embedding and hidden states are 128 and 512
  - PLM encoder: RoBERTa-base & BERT-base

https://github.com/2003pro/Graph2Tree/tree/master/math23k

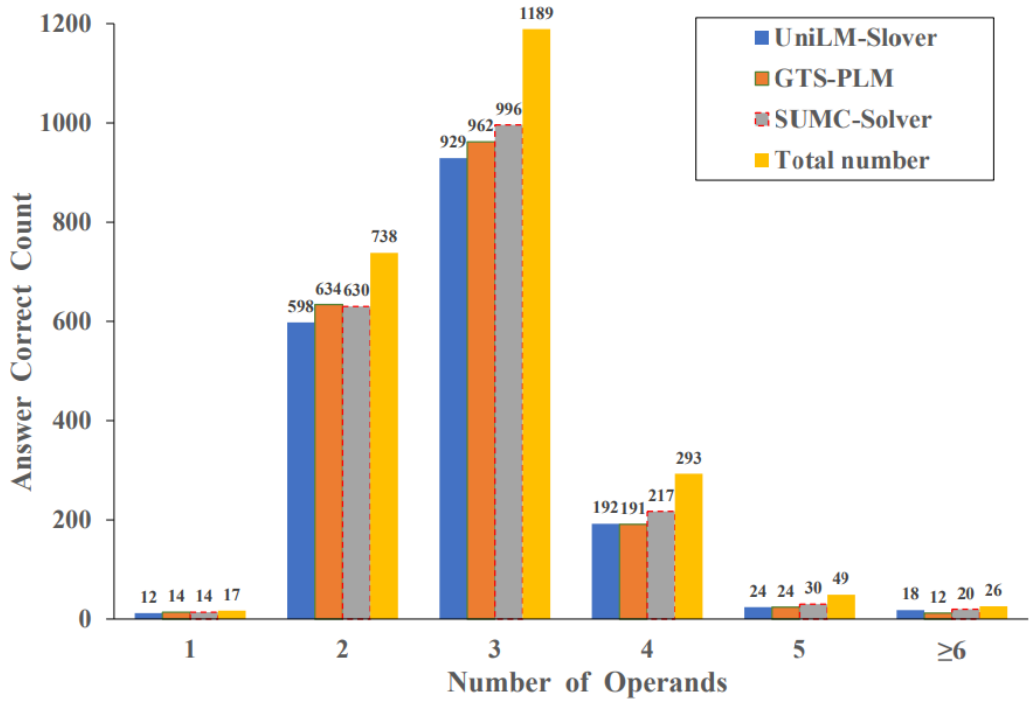
Experiment

- **Comparison in Low-resource Situations**
  - The annotation cost for MWPs is high, so it is desirable for the model to perform well in lower resource settings
  - SUMC-Solver consistently outperforms other models irrespective of the size of the training set



Experiment

- **Performance on Different Numbers of Operand**
  - Divide the test set (2,312 randomly sampled instances) into different levels according to the number of operands (numerical values in problems)
  - Outperform the baseline models on data requiring more operand
  - Our solver has the potential to solve more complex problems



Part 1.

Background

- **Today here to introduce MWP-NAS:**
  Non-autoregressive MWP solver (Seq2Exp)
  - A goal-driven manner for multi-branch decomposition to generate the MTree of expressions
  - A cross-goal attention strategy to pass information between goals during goal decomposing
  - Design two metrics based on MTree for better expression evaluation

Motivation			
A goal-driven manner	←	Seq2Tree	Goal-driven Tree-structured Model (GTS) (Xie and Sun, 2019)
Multi-branch decomposition	←	Seq2Exp	Structure-Unified M-Tree Coding Solver (SUMC-Solver) (Bin Wang et al., 2022)
Cross-goal Aattention	←	Prev Work	Non-autoregressive Transformer (Gu et al., 2018)
			Pointer Network (Peter Brown et al., 2018)

Previous Work

Non-autoregressive neural machine translation

ICLR 2018

Jiatao Gu†∗, James Bradbury‡ , Caiming Xiong‡ , Victor O.K. Li† & Richard Socher‡

‡Salesforce Research

†The University of Hong Kong

Introduction

- **A non-autoregressive translation model based on the Transformer network**
  - Some models avoid recurrence at train time by leveraging convolutions or self-attention
    - Still use of autoregressive decoding makes it impossible to take full advantage of parallelism during inference
  - Modify the encoder of the original Transformer network
    - Add a module that predicts fertilities:
      Sequences of numbers that form an important component of many traditional machine translation models(Brown et al., 1993)
  - Fertility
    - Supervised training
    - Provide the decoder at inference time with a globally consistent plan on which to condition its simultaneously computed outputs

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Background: Autoregressive Neural Machine Translation

- **Autoregressive Neural Machine Translation**
  - A model factors the distribution over possible output sentences into a chain of conditional probabilities with a left-to-right causal structure

$$p_{\mathcal{AR}}(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta)$$

  - Where the special tokens y_0 (e.g. $\langle \text{bos} \rangle$) and y_{T+1} (e.g. $\langle \text{eos} \rangle$) are used to represent the beginning and end of all target sentences

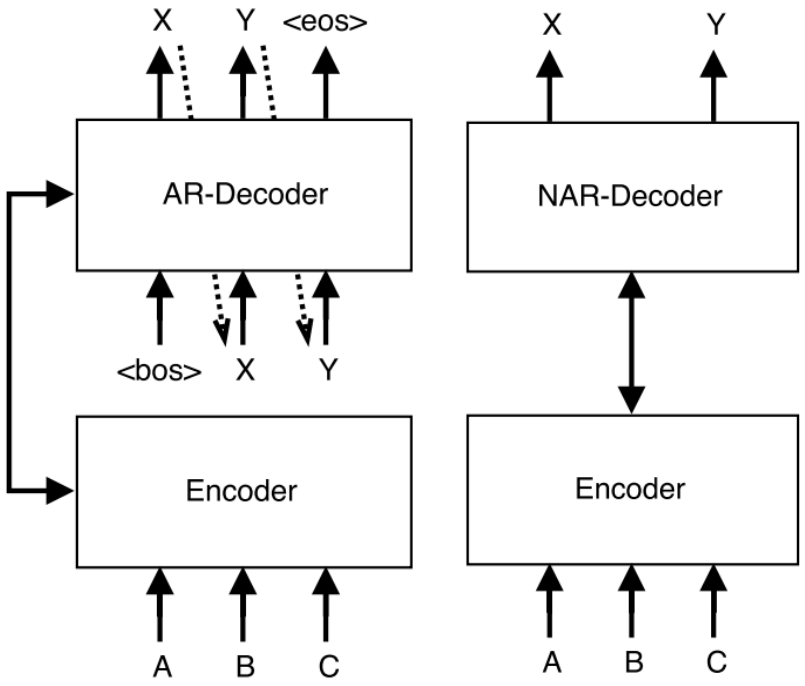
  - An encoder-decoder architecture (Sutskever et al., 2014) with a unidirectional RNN-based decoder is used to capture the causal structure of the output distribution

$$\mathcal{L}_{\text{ML}} = \log p_{\mathcal{AR}}(Y|X;\theta) = \sum_{t=1}^{T+1} \log p(y_t|y_{0:t-1}, x_{1:T'};\theta)$$

Ilya Sutskever et al. Sequence to sequence learning with neural networks. NIPS. 2014.

Background: Non-autoregressive Decoding

- **Pros and cons of autoregressive decoding**



Pros

- Effectively captures the distribution of real translations according to the word-by-word nature of human language production
- Easy to train with SOTA performance
- Beam search: an effective local search method for finding approximately-optimal output translations

Cons

- Sequential decoding:
  Autoregressive decoding prevents architectures from fully realizing train-time performance advantage during inference
- Beam search:
  - **Suffer from diminishing returns with respect to beam size**
  - **Exhibit limited search parallelism**
    because of computational dependence between beams

Background: Non-autoregressive Decoding

- **Towards non-autoregressive decoding**
  - Remove the autoregressive connection directly from an existing encoder-decoder model
  - Assumption: Target sequence length T can be modeled with a separate conditional distribution

$$p_{\mathcal{AR}}(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta) \Rightarrow p_{\mathcal{NA}}(Y|X;\theta) = p_L(T|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:T'};\theta)$$

Fertility Output

  - An explicit likelihood function: still use independent cross-entropy losses on each output distribution during training
  - Parallelism: These distributions can be computed in at inference time

Background: Non-autoregressive Decoding

- **Multimodality Problem: Conditional independence assumption**
  - The assumption prevents a model from properly capturing the highly multimodal distribution of target translations
  - Complete conditional independence assumption : Naive approach doesn't yield good results
    - Each token's distribution $p(y_t)$ depends only on the source sentence X
      - ➡ A poor approximation to the true target distribution
      - ➡ True target distribution exhibits strong correlation across time
    - Intuitively, such a decoder is akin to a panel of human translators
      - Ask to provide a single word independently of the words their colleagues choose

Source Sentence	Target distribution		
"Thank you." ➡	"Danke."	"Danke schön."	"Vielen Dank."

Source Sentence	A conditionally independent distribution		
"Thank you." ➡	"Danke."	"Danke Dank."	"Vielen schön."

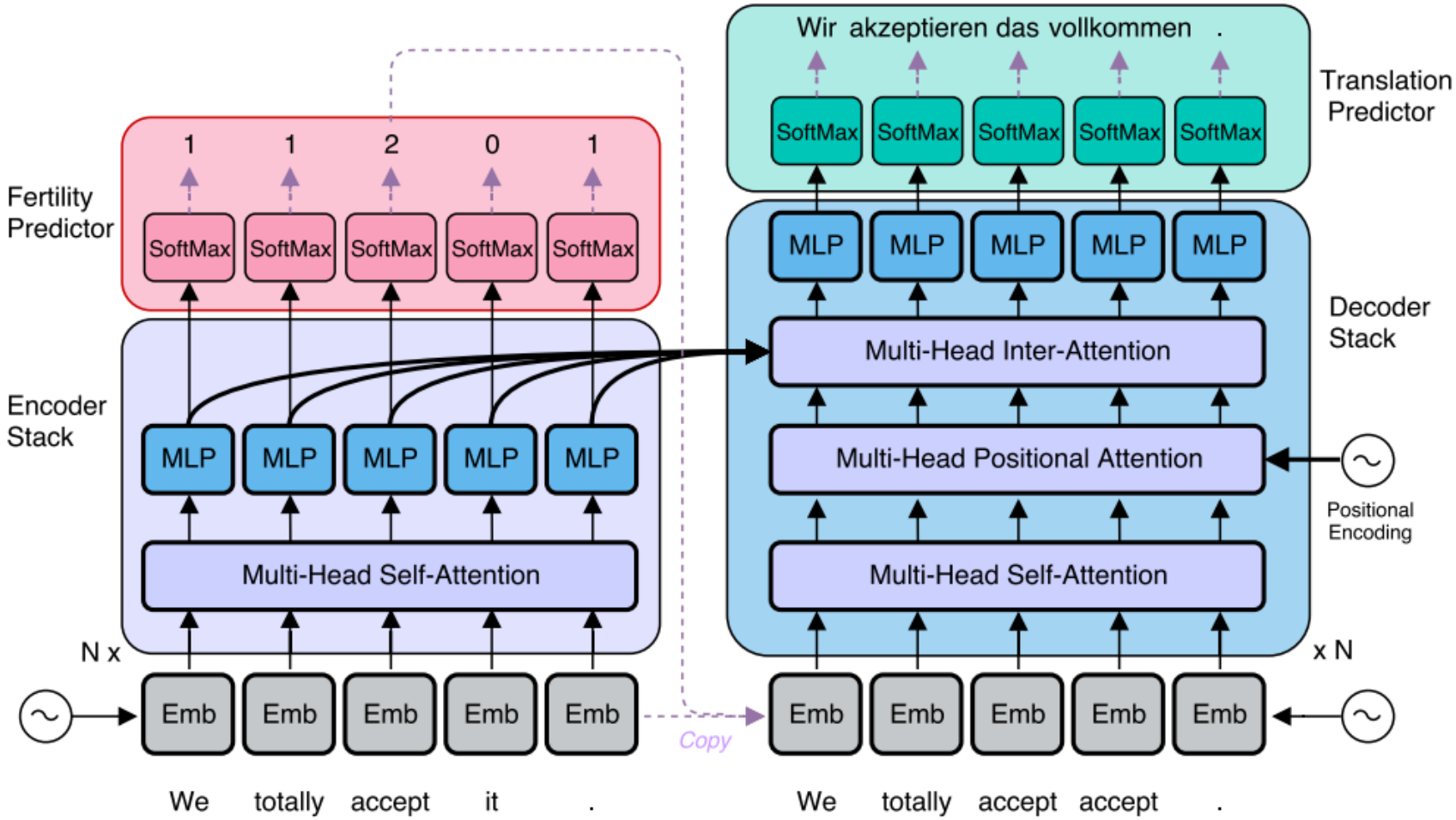
Vocabulary

- schön: 정말
- Vielen: 매우
- Danke: 감사합니다

The Non-autoregressive Transformer (NAT)

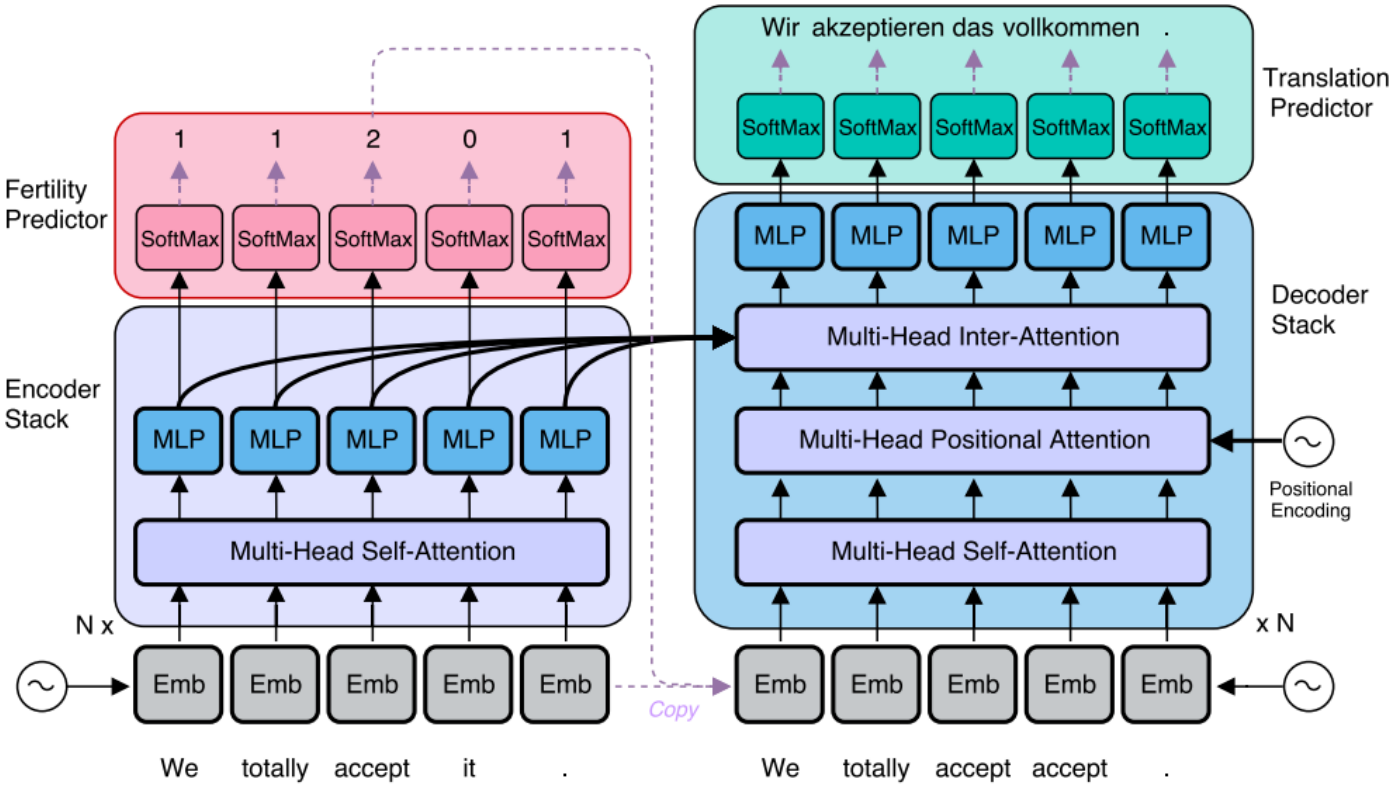
- **Multimodality Problem**

$$\mathcal{L}_{\text{ML}} \geq \mathop{\mathbb{E}}_{f_{1:T'} \sim q} \left(\underbrace{\sum_{t=1}^{T} \log p(y_t | x_1\{f_1\}, .., x_{T'}\{f_{T'}\}; \theta)}_{\text{Translation Loss}} + \underbrace{\sum_{t'=1}^{T'} \log p_F(f_{t'} | x_{1:T'}; \theta)}_{\text{Fertility Loss}} \right) + \mathcal{H}(q)$$



The Non-autoregressive Transformer (NAT)

- **Encoder & Decoder**



- **Encoder Stack**: Similar to the autoregressive Transformer
  - Both the encoder & decoder stacks are composed entirely of FFN (MLPs) & MHN modules
  - Since no RNNs are used, there is no inherent requirement for sequential execution
- **Decoder Inputs**: The first decoder layer
  - Cannot use time-shifted target outputs as the inputs (during training)
  - Cannot use previously predicted outputs as the inputs (during inference)
  - Non-causal self-attention: Avoid the causal mask used in the self-attention module

The Non-autoregressive Transformer (NAT)

- **Initialize the decoding process for Decoder Inputs**

  As the source and target sentences are often of different lengths

  - Copy source inputs uniformly:
    - Each decoder input t is a copy of the $\text{Round}(T't/T)$-th encoder input
    - Equivalent to "scanning" source inputs from left to right with a constant "speed"
    - Results in a decoding process that is deterministic given a (predicted) target length

  - Copy source inputs using fertilities (A more powerful way):
    - Copy each encoder input as a decoder input zero or more times, with the number of times each input is copied referred to as that input word's "fertility"
    - "scanning" source inputs from left to right with a "speed" that varies inversely with the fertility of each input
    - The decoding process is now conditioned on the sequence of fertilities

The Non-autoregressive Transformer (NAT)

- **Modeling Fertility to Tackle The Multimodality Problem**
  - Multimodality problem can be attacked by introducing a latent variable z to directly model the non-determinism in the translation process

One way to interpret this latent variable:

a sentence-level "plan" akin to those discussed in the language production literature (Martin et al., 2010)

- Desirable properties for this latent variable $p(y|x,z)$
  - Adding z to the conditioning context should account as much as possible for the correlations across time between different outputs
  - So that the remaining marginal probabilities at each output location are as close as possible to satisfying conditional independence

Randi Martin et al. Planning in sentence production: Evidence for the phrase as a default planning scope. Cognition, 116 (2):177–192, 2010.

The Non-autoregressive Transformer (NAT)

- **Modeling Fertility to Tackle The Multimodality Problem**
  - The factorization by length provides a very weak example of a latent variable model, satisfying the first and third property but not the first

$$p_{\mathcal{NA}}(Y|X;\theta) = p_L(T|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_{1:T'};\theta)$$

  - Propose the use of fertilities instead
    - The number of words in the source & target sentence can be aligned to that source word using a hard alignment algorithm like IBM Model 2 (Brown et al., 1993)

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

The Non-autoregressive Transformer (NAT)

- **Modeling Fertility to Tackle The Multimodality Problem**
  - One of the most important properties of the proposed NAT
    - Naturally introduces an informative latent variable
      when we choose to copy the encoder inputs based on predicted fertilities

$$p_{\mathcal{NA}}(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},..,x_{T'}\{f_{T'}\};\theta) \right)$$

$$\mathcal{F} = \{f_1,\ldots,f_{T'} | \textstyle\sum_{t'=1}^{T'} f_{t'} = T, f_{t'} \in \mathbb{Z}^*\}$$

Sentence Length

$l = 6, m = 7$

- The set of all fertility sequences -
  One fertility value per source word - that sum to the length of Y
- $x\{f\}$ denotes the token x repeated f times

Source Sentence

$e = And\ the\ programme\ has\ been\ implemented$

Target Setntence

$f = Le\ programme\ a\ ete\ mis\ en\ application$

The Non-autoregressive Transformer (NAT)

- **Alignment Models: IBM Model 2**
  - $t(f|e)$: Conditional probability of generating target word f from source word e
  - $q(j|i,l,m)$: Probability of alignment variable a_i (The value j , source & target l & m)

$$p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l, m) = \prod_{i=1}^{m} q(a_i|i,l,m) t(f_i|e_{a_i})$$

Alignment

Le	$\Rightarrow$	the
Programme	$\Rightarrow$	program
a	$\Rightarrow$	has
ete	$\Rightarrow$	been
mis	$\Rightarrow$	implemented
en	$\Rightarrow$	implemented
application	$\Rightarrow$	implemented

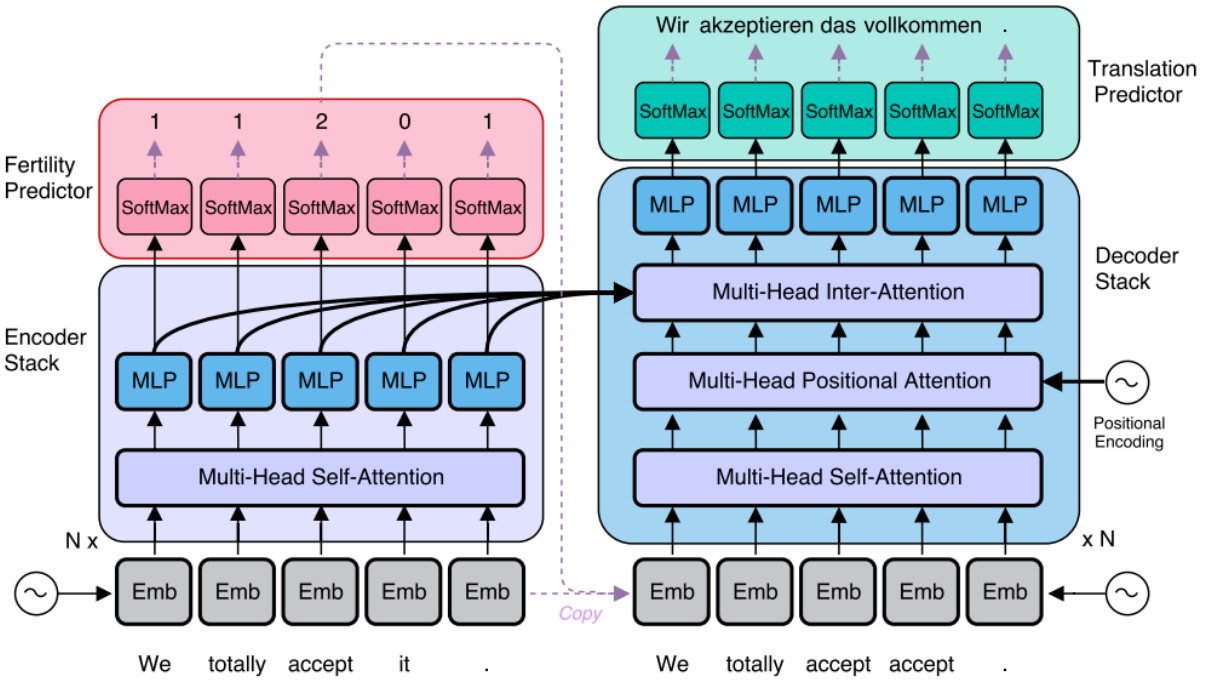
$$\begin{aligned}
& p(f_1 \dots f_m, a_1 \dots a_m | e_1 \dots e_l, m) \\
= \; & q(2|1,6,7) \times t(Le|the) \\
& \times q(3|2,6,7) \times t(Programme|program) \\
& \times q(4|3,6,7) \times t(a|has) \\
& \times q(5|4,6,7) \times t(ete|been) \\
& \times q(6|5,6,7) \times t(mis|implemented) \\
& \times q(6|6,6,7) \times t(en|implemented) \\
& \times q(6|7,6,7) \times t(application|implemented)
\end{aligned}$$

Michael Collins. Statistical Machine Translation: IBM Models 1 and 2. 2013.

The Non-autoregressive Transformer (NAT)

- **Modeling Fertility to Tackle The Multimodality Problem**

$$p_{\mathcal{NA}}(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},..,x_{T'}\{f_{T'}\};\theta) \right)$$



Fertility prediction

- Model the fertility $p_F(f_{t'}|x_{1:T'})$ at each position independently using a one-layer neural network
- Use a softmax classifier $(F = 50 \text{ in our experiments})$ on top of the output of the last encoder layer
- Fertility values are a property of each input word depend on information & context from the entire sentence

The Non-autoregressive Transformer (NAT)

- **Benefits of fertility as a latent variable**

$$p_{\mathcal{NA}}(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},..,x_{T'}\{f_{T'}\};\theta) \right)$$

  - An external aligner provides a simple and fast approximate inference model that effectively reduces the unsupervised training problem to two supervised ones
  - Using fertilities as a latent variable: Solve the multimodality problem by providing a natural factorization of the output space
    - The global choice of mode is factored into a set of local mode choices: namely, how to translate each input word
  - Including both fertilities and reordering in the latent variable would provide complete alignment statistics

Translation Predictor And The Decoding Process

- **Benefits of fertility as a latent variable**

$$Y = G(x_{1:T'}, f_{1:T'}; \theta)$$

  - Represent the optimal translation given a source sentence and a sequence of fertility values
  - But searching and marginalizing over the whole fertility space is still intractable
  - Propose three heuristic decoding algorithms to reduce the search space of the NAT model

Translation Predictor And The Decoding Process

- **Heuristic decoding algorithms to reduce the search space of the NAT model**

$$\hat{Y}_{\text{argmax}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta) \text{ where } \hat{f}_{t'} = \underset{f}{\text{argmax}}\, p_F(f_{t'}|x_{1:T'}; \theta)$$

Argmax decoding

- The fertility sequence is modeled with a conditionally independent factorization
- So simply estimate the best translation by choosing the highest-probability fertility for each input word

$$\hat{Y}_{\text{average}} = G(x_{1:T'}, \hat{f}_{1:T'}; \theta) \text{ where } \hat{f}_{t'} = \text{Round}\left(\sum_{f_{t'}=1}^{L} p_F(f_{t'}|x_{1:T'}; \theta) f_{t'}\right)$$

Average decoding

- Estimate each fertility as the expectation of its corresponding softmax distribution

Translation Predictor And The Decoding Process

- **Heuristic decoding algorithms to reduce the search space of the NAT model**

The autoregressive teacher to identify the best overall translation

$$\hat{Y}_{\text{NPD}} = G(x_{1:T'}, \underset{f_{t'} \sim p_F}{\operatorname{argmax}}\, p_{\mathcal{AR}}(G(x_{1:T'}, f_{1:T'}; \theta)|X; \theta); \theta)$$

Noisy parallel decoding (NPD)

- A more accurate approximation of the true optimum of the target distribution
  - Inspired by Cho (2016), draw samples from the fertility space and compute the best translation for each fertility sequence
- A scoring function: it can run as fast as it does at train time because it can be provided with all decoder inputs in parallel
- A stochastic search method
- Increase the computational resources required linearly by the sample size
  - The process only doubles the latency compared to computing a single translation if sufficient parallelism is available

Kyunghyun Cho. Noisy parallel approximate decoding for conditional recurrent language model. arXiv preprint arXiv:1605.03835, 2016.

Training

- **A variational bound for the overall maximum likelihood loss**

$$\mathcal{L}_{\mathrm{ML}} = \log p_{\mathcal{NA}}(Y|X;\theta) = \log \sum_{f_{1:T'} \in \mathcal{F}} p_F(f_{1:T'}|x_{1:T'};\theta) \cdot p(y_{1:T}|x_{1:T'}, f_{1:T'};\theta)$$

$$\geq \mathop{\mathbb{E}}_{f_{1:T'} \sim q} \left(\underbrace{\sum_{t=1}^{T} \log p(y_t|x_1\{f_1\}, .., x_{T'}\{f_{T'}\};\theta)}_{\text{Translation Loss}} + \underbrace{\sum_{t'=1}^{T'} \log p_F(f_{t'}|x_{1:T'};\theta)}_{\text{Fertility Loss}} \right) + \mathcal{H}(q)$$

- A discrete sequential latent variable
- Conditional posterior distribution
- Approximate using a proposal distribution

$f_{1:T'}$

$q(f_{1:T'}|x_{1:T'}, y_{1:T})$

$p(f_{1:T'}|x_{1:T'}, y_{1:T};\theta)$

Training

- **A variational bound for the overall maximum likelihood loss**

$$\mathcal{L}_{\mathrm{ML}} \geq \mathop{\mathbb{E}}_{f_{1:T'} \sim q} \left(\underbrace{\sum_{t=1}^{T} \log p(y_t | x_1\{f_1\}, .., x_{T'}\{f_{T'}\}; \theta)}_{\text{Translation Loss}} + \underbrace{\sum_{t'=1}^{T'} \log p_F(f_{t'} | x_{1:T'}; \theta)}_{\text{Fertility Loss}} \right) + \mathcal{H}(q)$$

  - Simplifies the inference process considerably, as the expectation over q is deterministic
    - Choose a proposal distribution q defined by a separate, fixed fertility model
    - Possible options include the output of an external aligner, which produces a deterministic sequence of integer fertilities in our fixed autoregressive teacher model
  - Allow us to train the entire model in a supervised fashion, using the inferred fertilities to simultaneously train the translation model p and supervise the fertility neural network model p_F

Training

- **Previous work: sequence-level knowledge distillation (Kim & Rush, 2016)**
  - Apply the distillation to construct a new corpus by training an autoregressive machine translation model, known as the teacher, on an existing training corpus
  - Use that model's greedy outputs as the targets for training the non-autoregressive student
  - The resulting targets are less noisy and more deterministic, as the trained model will consistently translate a sentence like "Thank you." into the same German translation every time

Training

- **Fine-Tuning**
  - Some drawbacks from a decomposition of the overall maximum likelihood loss into translation and fertility terms
    - Heavily relies on the deterministic, approximate inference model provided by the external alignment system
  - It would be desirable to train entire model, including the fertility predictor, end to end

Additional loss term consisting of the reverse K-L divergence with the teacher model (word-level knowledge distillation)

$$\mathcal{L}_{\mathrm{RKL}}\left(f_{1:T'};\theta\right)=\sum_{t=1}^{T}\sum_{y_t}\left[\log p_{\mathcal{AR}}\left(y_t|\hat{y}_{1:t-1},x_{1:T'}\right)\cdot p_{\mathcal{NA}}\left(y_t|x_{1:T'},f_{1:T'};\theta\right)\right] \qquad \hat{y}_{1:T}=G(x_{1:T'},f_{1:T'};\theta)$$

$$\mathcal{L}_{\mathrm{FT}}=\lambda\left(\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim p_F}\left(\mathcal{L}_{\mathrm{RKL}}\left(f_{1:T'}\right)-\mathcal{L}_{\mathrm{RKL}}\left(\bar{f}_{1:T'}\right)\right)}_{\mathcal{L}_{\mathrm{RL}}}+\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim q}\left(\mathcal{L}_{\mathrm{RKL}}\left(f_{1:T'}\right)\right)}_{\mathcal{L}_{\mathrm{BP}}}\right)+\left(1-\lambda\right)\mathcal{L}_{\mathrm{KD}}$$

  - Such a loss is more favorable towards highly peaked student output distributions than a standard cross-entropy error would be

Training

- **Fine-Tuning**

Additional loss term consisting of the reverse K-L divergence with the teacher model (word-level knowledge distillation)

$$\mathcal{L}_{\text{RKL}}\left(f_{1:T'};\theta\right)=\sum_{t=1}^{T}\sum_{y_t}\left[\log p_{\mathcal{AR}}\left(y_t|\hat{y}_{1:t-1},x_{1:T'}\right)\cdot p_{\mathcal{NA}}\left(y_t|x_{1:T'},f_{1:T'};\theta\right)\right] \qquad \hat{y}_{1:T}=G(x_{1:T'},f_{1:T'};\theta)$$

$$\mathcal{L}_{\text{FT}}=\lambda\left(\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim p_F}\left(\mathcal{L}_{\text{RKL}}\left(f_{1:T'}\right)-\mathcal{L}_{\text{RKL}}\left(\bar{f}_{1:T'}\right)\right)}_{\mathcal{L}_{\text{RL}}}+\underbrace{\mathop{\mathbb{E}}_{f_{1:T'}\sim q}\left(\mathcal{L}_{\text{RKL}}\left(f_{1:T'}\right)\right)}_{\mathcal{L}_{\text{BP}}}\right)+\left(1-\lambda\right)\mathcal{L}_{\text{KD}}$$

$$\bar{f}_{1:T'}=\hat{Y}_{\text{average}}=G(x_{1:T'},\hat{f}_{1:T'};\theta)\ \text{where}\ \hat{f}_{t'}=\text{Round}\left(\sum_{f_{t'}=1}^{L}p_F(f_{t'}|x_{1:T'};\theta)f_{t'}\right)$$

- Train the whole model jointly with a weighted sum of the original distillation loss and two such terms, one an expectation over the predicted fertility distribution, normalized with a baseline, and the other based on the external fertility inference model

Experiment

- **Datasets**
  - IWSLT16 En–De2
    - Use separate English and German vocabulary and embeddings
  - WMT14 En–De,3 and WMT16 En–
    - Use shared BPE vocabulary and additionally share encoder and decoder word embeddings

- **Teacher**
  - Sequence-level knowledge distillation is applied to alleviate multimodality in the training dataset, using autoregressive models as the teacher
  - The same teacher model used for distillation is also used as a scoring function for fine-tuning and noisy parallel decodig
  - Transformer architecture
    - Use the same sizes and hyperparameters for each student and its respective teacher, with the exception of the newly added positional self-attention and fertility prediction module

Part 7.

Experiment

- **Results**
  - The NAT performs between 2-5 BLEU points worse than its autoregressive teacher
  - NPD improves the performance of our non-autoregressive model
  - shows a speedup of more than a factor of 10 over greedy autoregressive decoding
  - Latencies for decoding with NPD, regardless of sample size, could be reduced to about 80ms by parallelizing across multiple GPUs because

BLEU score & Latency

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Latency is computed as the time to decode a single sentence without minibatching, averaged over the whole test set

Experiment

- **Ablation Study**
  - Fine-tuning doesn't converge with reinforcement learning alone, or with $\mathcal{L}_{\mathrm{BP}}$
  - Use of all three fine-tuning terms together leads to an improvement
  - Training the student model from a distillation corpus produced using beam search is similar to training from the greedily-distilled corpus
  - The translations produced by the NAT with NPD are also noticeably more literal.
    - Instances of repeated words or phrases, highlighted in gray, are most prevalent in the non-autoregressive output for the relatively complex first example sentence
    - A pair in the second, are not present in the versions with noisy parallel decoding
    - NPD scoring using the teacher model can filter out such mistakes

Source:	politicians try to pick words and use words to shape reality and control reality , but in fact , reality changes words far more than words can ever change reality .
Target:	Politiker versuchen Worte zu benutzen , um die Realität zu formen und die Realität zu kontrollieren , aber tatsächlich verändert die Realität Worte viel mehr , als Worte die Realität jemals verändern könnten .
AR:	Politiker versuchen Wörter zu wählen und Wörter zur Realität zu gestalten und Realität zu steuern , aber in Wirklichkeit verändert sich die Realität viel mehr als Worte , die die Realität verändern können .
NAT:	Politiker versuchen , Wörter wählen und zu verwenden , um Realität zu formen und Realität zu formen , aber tatsächlich ändert Realität Realität viel mehr als Worte die Realität Realität verändern .
NAT+NPD:	Politiker versuchen , Wörter wählen und zu verwenden , um Realität Realität formen und die Realität zu formen , aber tatsächlich ändert die Realität Worte viel mehr als Worte jemals die Realität verändern können .

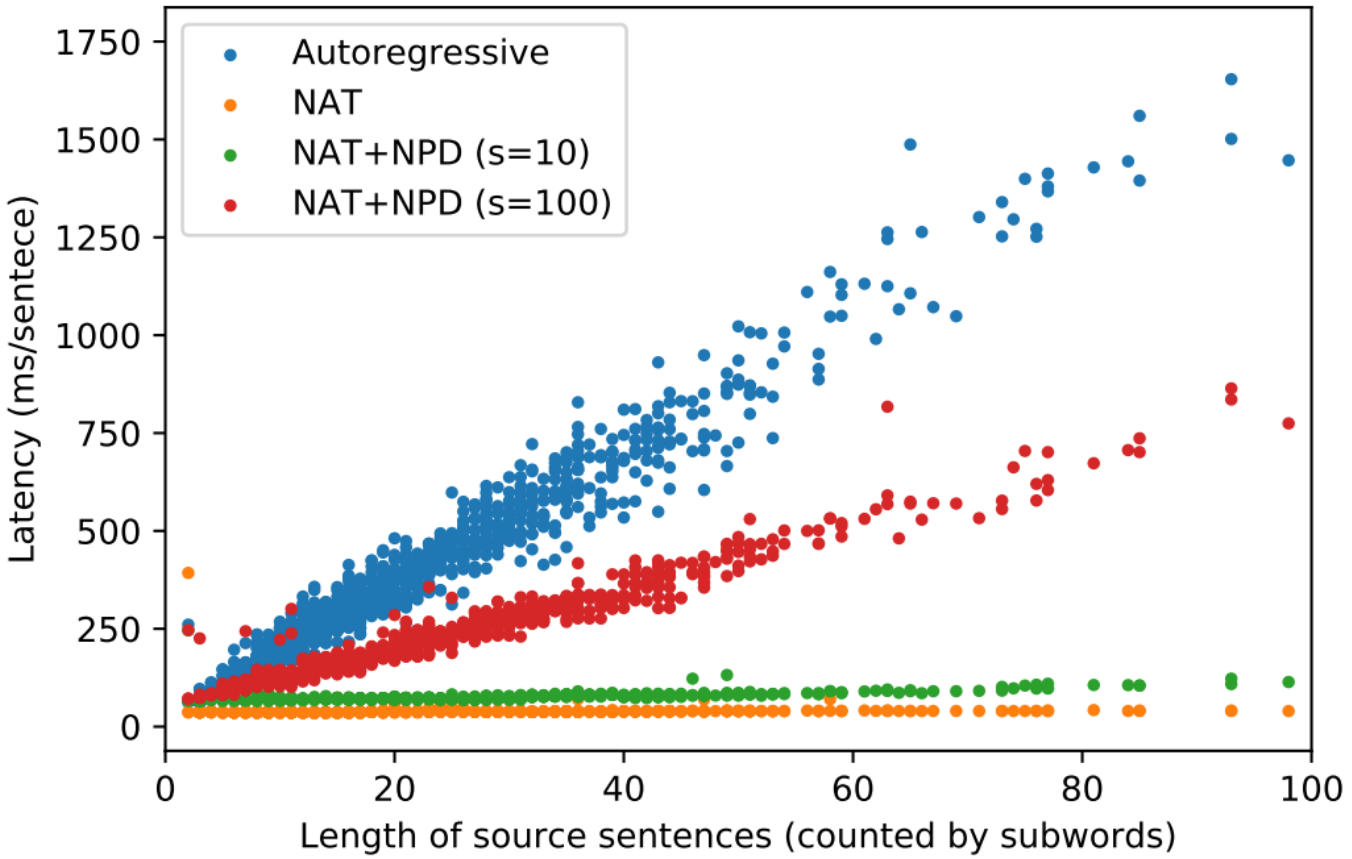
Experiment

- **Ablation Study**
  - Noisy parallel decoding process demonstrates the diversity of translations that can be found by sampling from the fertility space
    - Left: Sampled fertility sequences from the encoder, represented with their corresponding decoder input sequences
    - Right: Values for the latent variable leads to a different possible output translation
    - Transformer picks the best translation (red), a process which is faster than directly using it to generate output

se lucreaza la solutii de genul acesta .	
se la solutii de genul acesta .	solutions on this kind are done .
se lucreaza la solutii de acesta .	work done on solutions like this .
se lucreaza solutii de genul acesta .	solutions on this kind is done .
se se lucreaza la solutii de acesta .	work is done on solutions like this .
se lucreaza lucreaza la solutii de acesta .	work is done on solutions like this .
se se lucreaza lucreaza la solutii de acesta .	**work is being done on solutions like this .**
se se lucreaza lucreaza la solutii de de acesta .	work is being done on solutions such as this .
se se lucreaza lucreaza la solutii de genul acesta .	work is being done on solutions such this kind .

Appendix

- **The translation latency for a single sentence**
  - When using NPD with sample size 100, the level of parallelism is enough to more than saturate the GPU, leading again to linear latencies



Part 1.

Background

- **Today here to introduce MWP-NAS:**
  Non-autoregressive MWP solver (Seq2Exp)
  - A goal-driven manner for multi-branch decomposition to generate the MTree of expressions
  - A cross-goal attention strategy to pass information between goals during goal decomposing
  - Design two metrics based on MTree for better expression evaluation

Motivation			
A goal-driven manner	←	Seq2Tree	Goal-driven Tree-structured Model (GTS) (Xie and Sun, 2019)
Multi-branch decomposition	←	Seq2Exp	Structure-Unified M-Tree Coding Solver (SUMC-Solver) (Bin Wang et al., 2022)
Cross-goal Aattention	←	Prev Work	Non-autoregressive Transformer (Gu et al., 2018)
			Pointer Network (Peter Brown et al., 2018)

Previous Work

Pointer Networks

NeurIPS 2015

Oriol Vinyals*, Meire Fortunato*, Navdeep Jaitly†

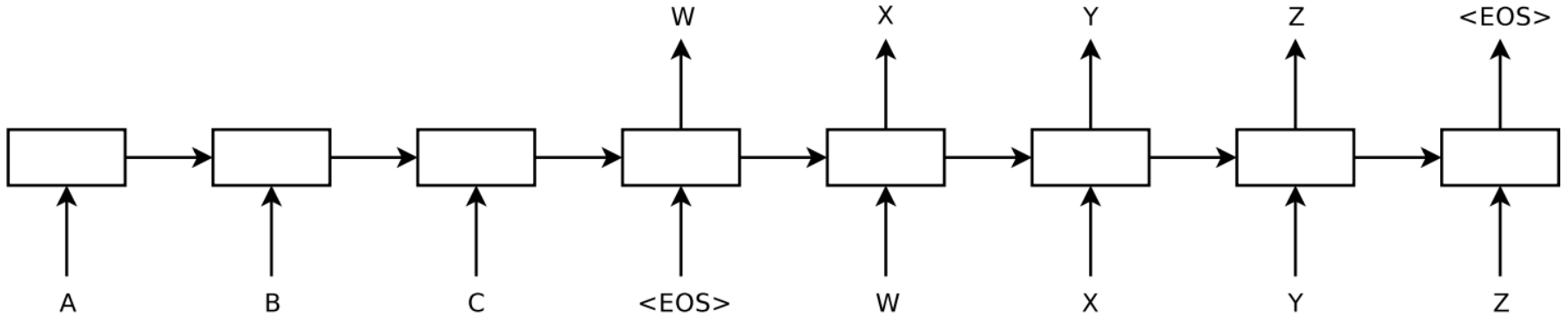
***Google Brain**

†Department of Mathematics, UC Berkeley

Introduction

- **Recurrent Neural Networks (RNNs)**
  - This architecture limited them to settings where the inputs and outputs were available at a fixed frame rate

- **Sequence-to-sequence paradigm**
  - Remove these constraints by using one RNN to map an input sequence to an embedding and another (possibly the same) RNN to map the embedding to an output sequence

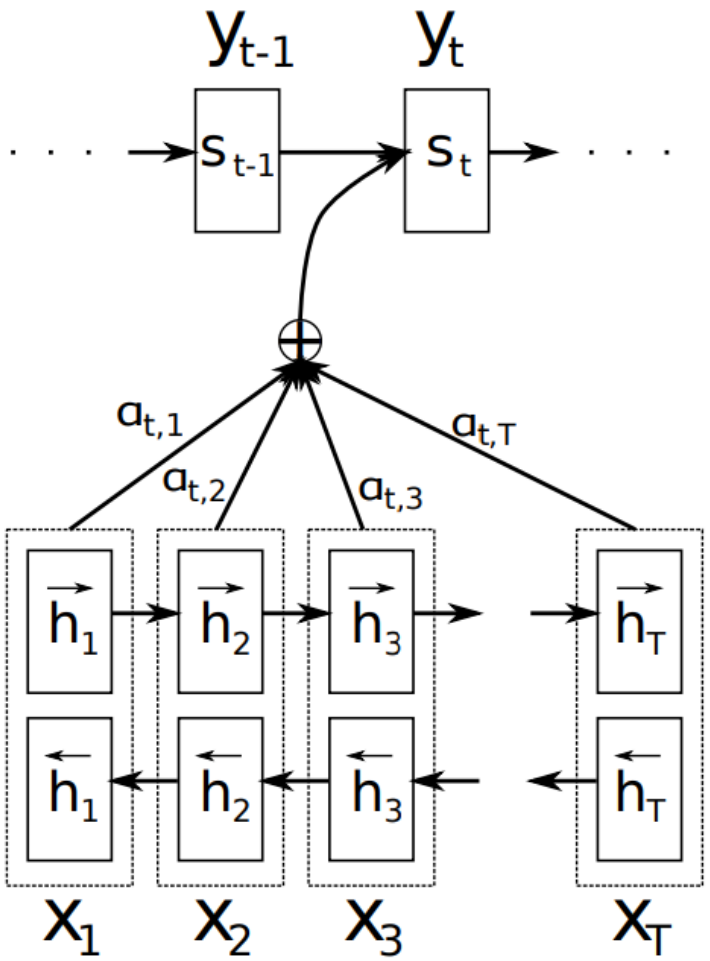


Introduce many short term dependencies in the data that make the optimization problem much easier

Ilya Sutskever et al. Sequence to sequence learning with neural networks. NIPS. 2014.

Introduction

- **Bahdanau et. al., 2015**

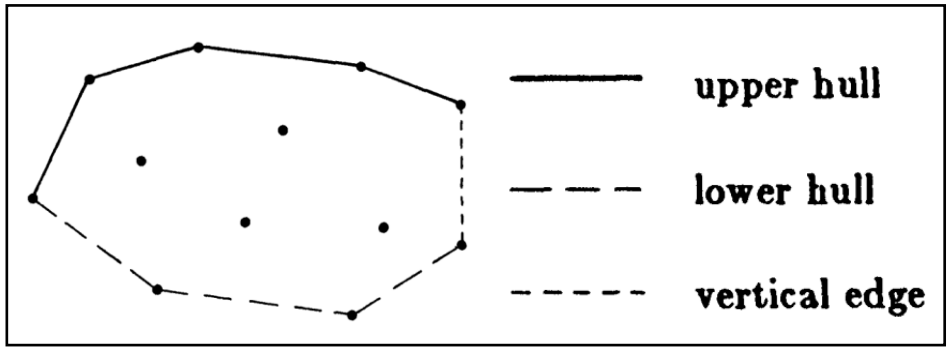


- Augment the decoder by propagating extra contextual information from the input using a content-based attentional mechanism
  - Make it possible to apply RNNs to new domains, achieving SOTA results in NLP
- Still require the size of the output dictionary to be fixed a priori
  - Cannot directly apply this framework to combinatorial problems where the size of the output dictionary depends on the length of the input sequence

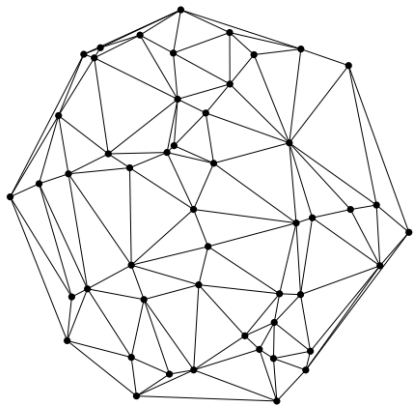
Dzmitry Bahdanau et al. Neural machine translation by jointly learning to align and translate. ICLR. 2015.

Introduction

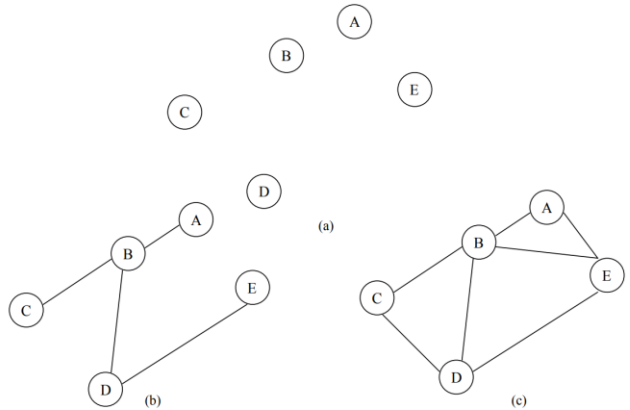
- **Address this limitation**
  - Repurpose the attention mechanism of [5] to create pointers to input elements
  - Proposed Method: Pointer Networks (Ptr-Nets) with combinatorial optimization problems
    - Computing planar convex hulls
    - Delaunay triangulations
    - The symmetric planar Travelling Salesman Problem (TSP)
  - Models produce approximate solutions to these problems in a purely data drive fashion



Convex Hull



Delaunay triangulation



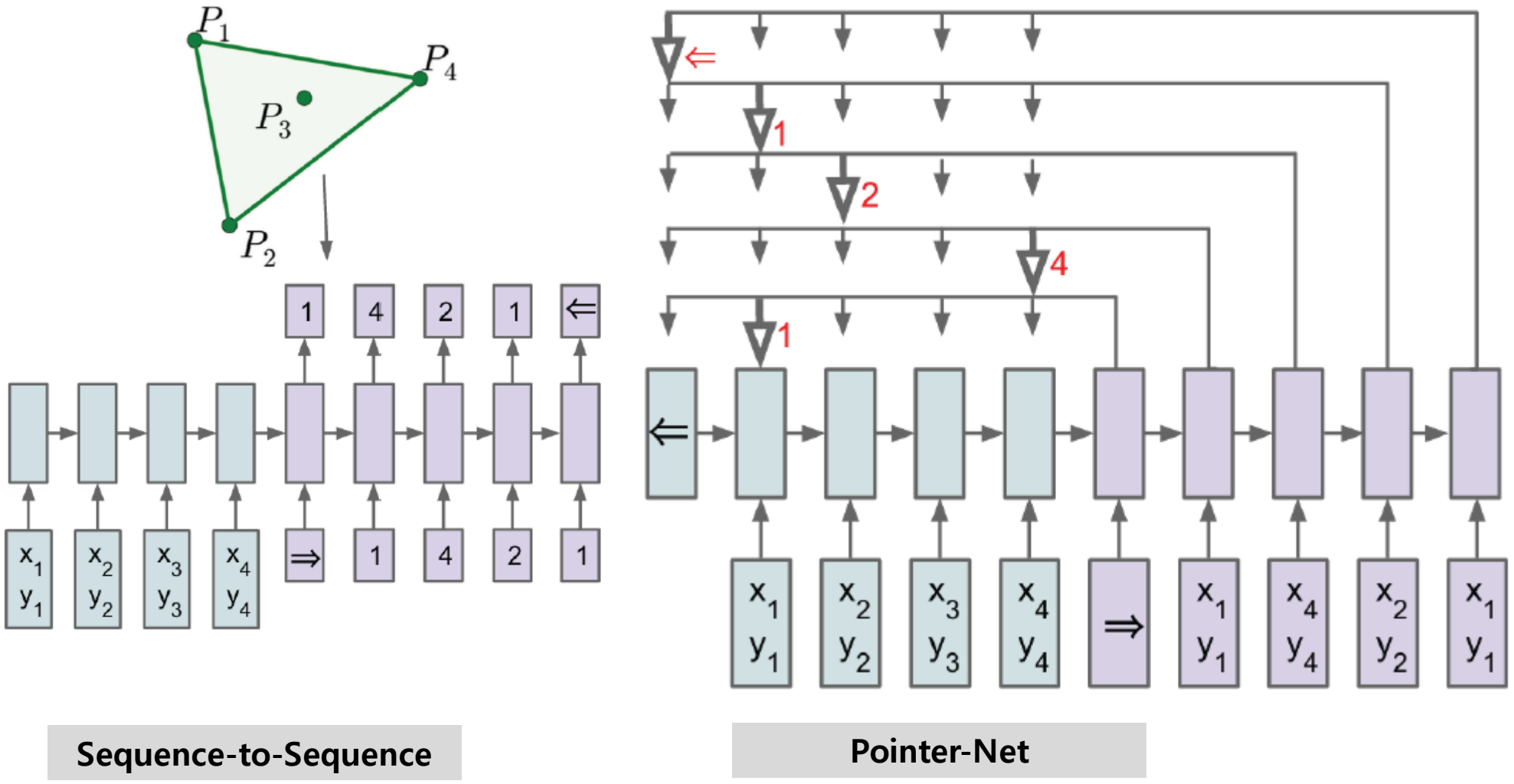
The Traveling Salesman Problem

David G. Kirkpatrick et al. The Ultimate Planar Convex Hull Algorithm? SIAM J. COMPUT. 1986
Prof. Emo Welzl and Prof. Bernd Gärtner. Theory of Combinatorial Algorithms Chapter 6 Delaunay Triangulations. Swiss Federal Institue of Technology Zurich. 2013.
Prof. Ioannis Tollis. HY-583 Graph Algorithms Chapter 10 The Traveling Salesman Problem. University of Crete. 2021.

Introduction

- **Contribution**
  - Pointer Net is simple and effective
    - Deal with the fundamental problem of representing variable length dictionaries by using a softmax probability distribution as a "pointer"
  - Distinct non-trivial algorithmic problems involving geometry
    - The learned model generalizes to test problems with more points than the training problems
  - Learns a competitive small scale $(n \leq 50)$ TSP approximate solver
    - A purely data driven approach can learn approximate solutions to problems that are computationally intractable

Introduction



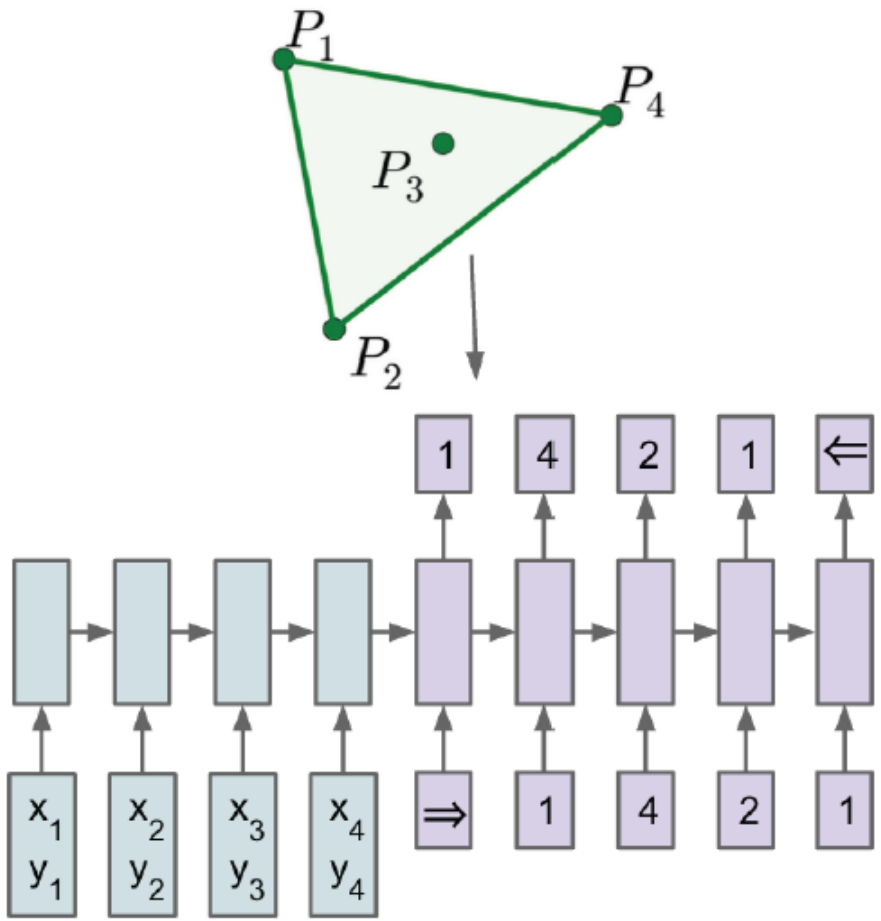
Sequence-to-Sequence

Pointer-Net

Peter Brown et al. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263–311, 1993.

Sequence-to-Sequence Model

- **The conditional probability**



$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$

- A training pair $(\mathcal{P},\mathcal{C}^{\mathcal{P}})$
- Use a parametric model (an RNN with parameters θ)
  - Estimate the terms of the probability chain rule
- A sequence of n vectors

$$\mathcal{P} = \{P_1,\ldots,P_n\}$$

- A sequence of $m(\mathcal{P})$ indices, each between 1 and n

$$\mathcal{C}^{\mathcal{P}} = \{C_1,\ldots,C_{m(\mathcal{P})}\}$$

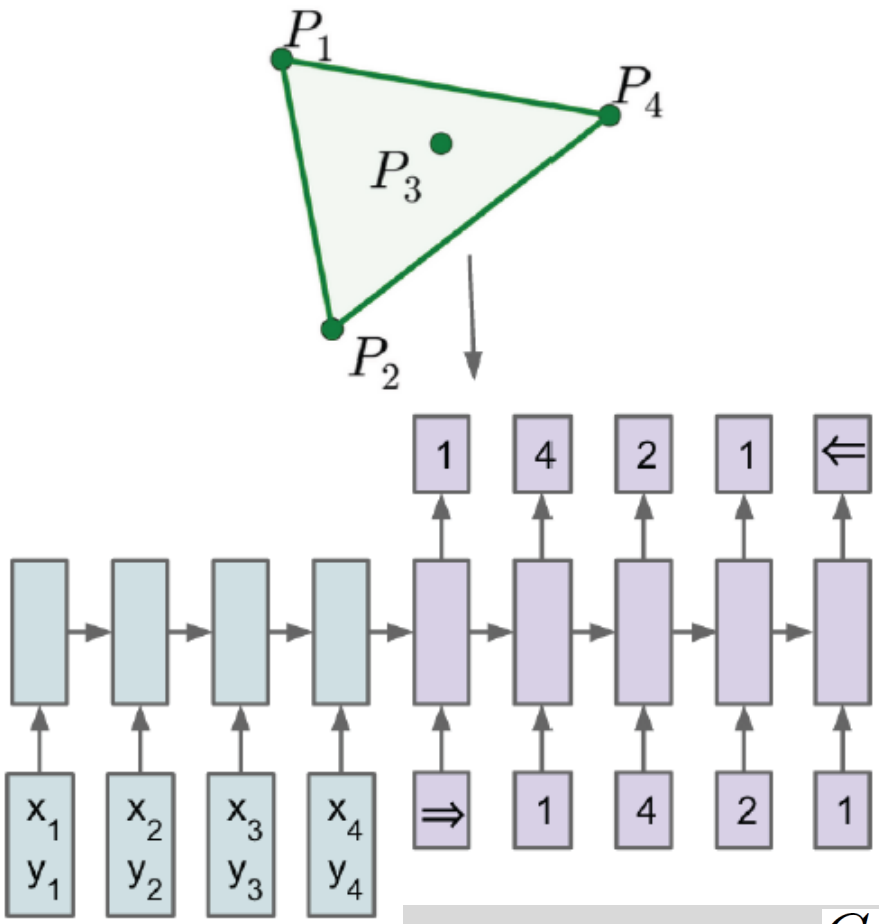
- Training Stage: Maximize the conditional probabilities

$$\theta^* = \arg\max_\theta \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)$$

Sequence-to-Sequence Model

- **The conditional probability**

$$p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta) = \prod_{i=1}^{m(\mathcal{P})} p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$



◦ Use an Long Short Term Memory (LSTM)

$$p_\theta(C_i|C_1,\ldots,C_{i-1},\mathcal{P};\theta)$$

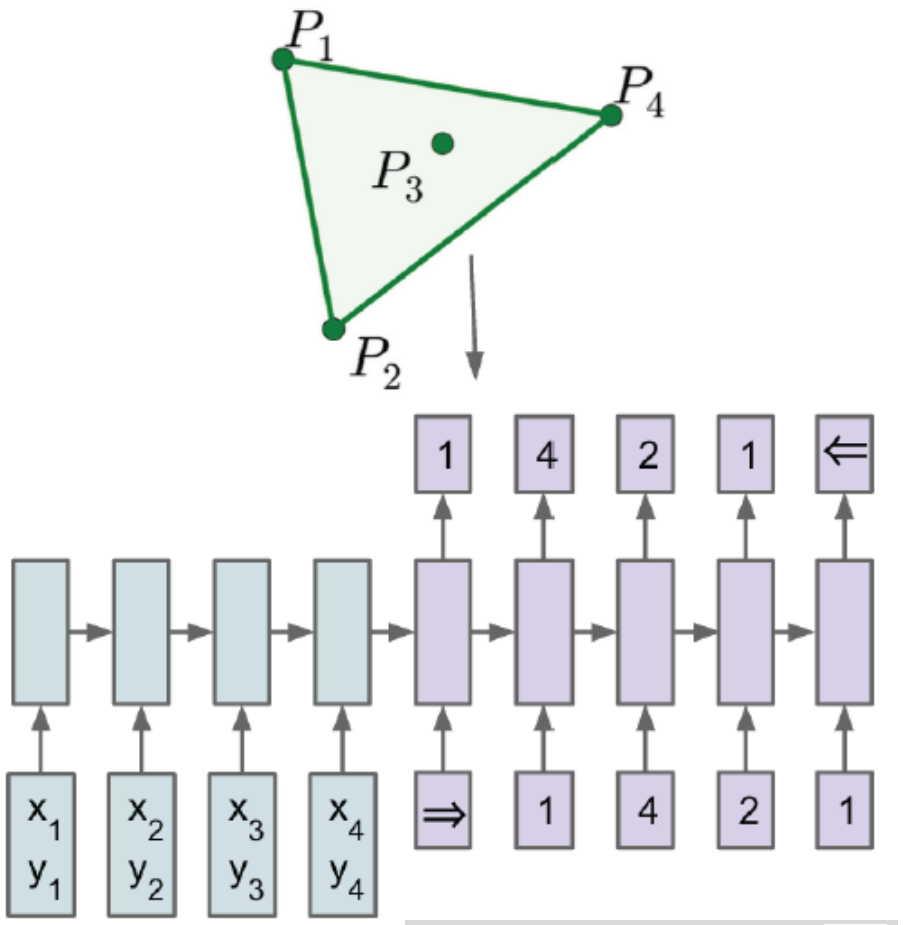
◦ The RNN is fed P_i at each time step, i , until the end of the input sequence is reached, at which time a special symbol, $\Rightarrow$ (input)

◦ Switches to the generation mode until the network encounters the special symbol, $\Leftarrow$ (termination)

No statistical independence assumptions
(The former, The latter RNN)

Sequence-to-Sequence Model

- **Sequence-to-sequence model**



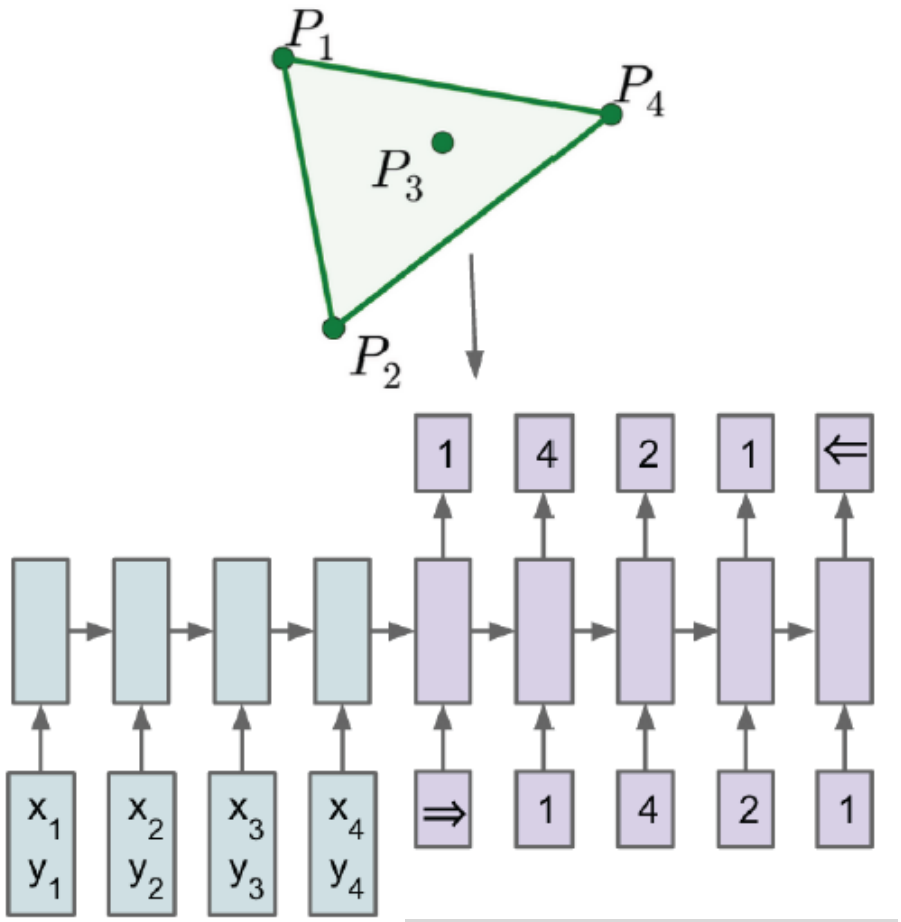
Training $$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)$$

Inference $$\hat{\mathcal{C}}^{\mathcal{P}} = \arg\max_{\mathcal{C}^{\mathcal{P}}} p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta^*)$$

- A sequence $\mathcal{P}$
- The learnt parameters θ^*
  - Select the sequence $\hat{\mathcal{C}}^{\mathcal{P}}$ with the highest probability
- Find the optimal sequence $\hat{\mathcal{C}}$
  - Problem - Computationally impractical:
    Combinatorial number of possible output sequences
  - Solution: Use a beam search procedure to find the best possible sequence given a beam size

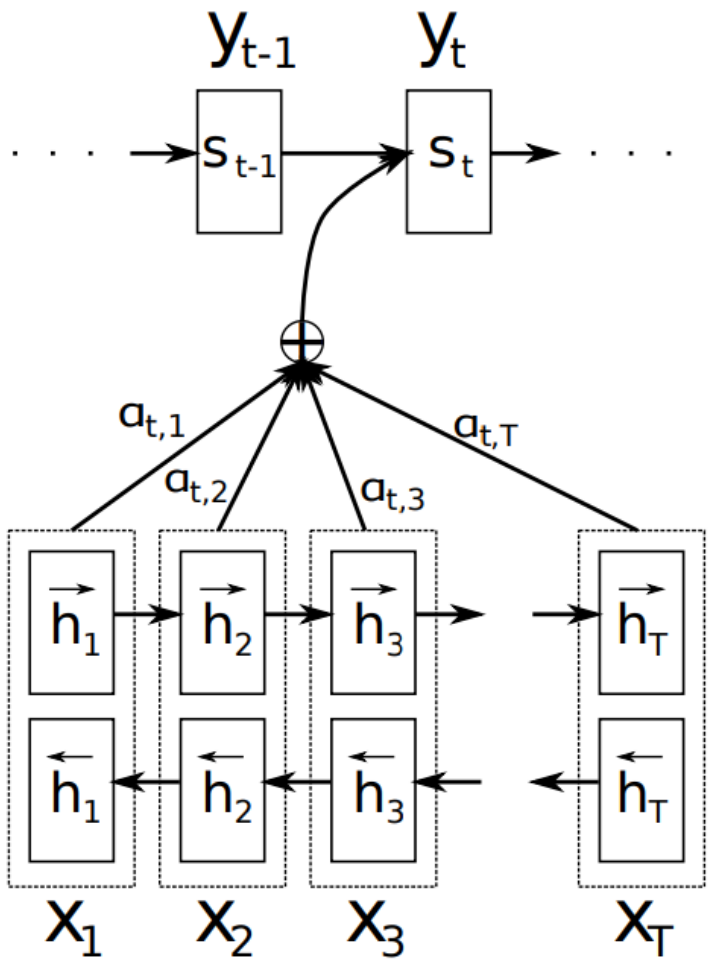
Sequence-to-Sequence Model

- **Training & Inference**



- The output dictionary size for all symbols C_i
  - It is fixed and equal to n, since the outputs are chosen from the input
- Need to train a separate model for each n
  - Prevents us from learning solutions to problems that have an output dictionary with a size that depends on the input sequence length
- The number of outputs $O(n)$
- Computational complexity $O(n)$
- Exact algorithms for the problems are more costly
  - $O(n \log n)$ For the convex hull problem

Content Based Input Attention



- **Vanilla sequence-to-sequence model**
  - Produce the entire output sequence $\mathcal{C}^{\mathcal{P}}$
    - Use the fixed dimensional state of the recognition RNN at the end of the input sequence $\mathcal{P}$
  - Constraint
    - Limit the amount of information and computation that can flow through to the generative model

- **Attention model of (Bahdanau et. al., 2015)**
  - Ameliorate this problem
    - Augment the encoder and decoder RNNs with an additional NN that uses an attention mechanism over the entire sequence of encoder RNN states

Dzmitry Bahdanau et al. Neural machine translation by jointly learning to align and translate. ICLR. 2015.

Content Based Input Attention

- **Compute the attention vector at each output time i**

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) && j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) && j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

  - LSTM RNN: Use the state after the output gate has been component-wise multiplied by the cell activations
  - Softmax normalizes the vector u^i to be the "attention" mask over the inputs
  - Learnable parameters of the model: **A Vector** v **/ Square matrices** $W_1 W_2$
  - d_i' d_i are concatenated and used as the hidden states
    - Hidden states from which we make predictions and which we feed to the next time step in the recurrent model

Content Based Input Attention

- **Compute the attention vector at each output time i**

$$
\begin{aligned}
u_j^i &= v^T \tanh(W_1 e_j + W_2 d_i) && j \in (1, \ldots, n) \\
a_j^i &= \text{softmax}(u_j^i) && j \in (1, \ldots, n) \\
d_i' &= \sum_{j=1}^{n} a_j^i e_j
\end{aligned}
$$

  - Computational complexity $O(n^2)$
  - Performs significantly better than the seq2-to-seq2 model on the convex hull problem
  - Not applicable to problems where the output dictionary size depends on the input
  - Nevertheless, a very simple extension (or rather reduction) of the model allows us to do this easily

Ptr-Net (Pointer-Net)

- **Modification of the attention model**

$$u_j^i = v^T \tanh(W_1 e_j + W_2 d_i) \quad \Rightarrow \quad u_j^i = v^T \tanh(W_1 e_j + W_2 d_i)$$

$$a_j^i = \text{softmax}(u_j^i) \quad d'_i = \sum_{j=1}^{n} a_j^i e_j \quad \Rightarrow \quad p(C_i|C_1, \ldots, C_{i-1}, \mathcal{P}) = \text{softmax}(u^i)$$

  - Solve combinatorial optimization problems
    - Output dictionary size depends on the number of elements in the input sequence
    - Use a softmax distribution over a fixed sized output dictionary to compute $p(C_i|C_1, \ldots, C_{i-1}, \mathcal{P})$
  - Don't blend the encoder state e_j to propagate extra information to the decoder
  - Instead, use u_j^i as pointers to the input elements
  - Targets problems whose outputs are discrete and correspond to positions in the input
  - At inference, Don't respect the constraint that the outputs map back to the inputs exactly
  - Without constraints, the predictions are bound to become blurry over longer sequence

Part 1.

Background

- **Today here to introduce MWP-NAS:**
  Non-autoregressive MWP solver (Seq2Exp)
  - A goal-driven manner for multi-branch decomposition to generate the MTree of expressions
  - A cross-goal attention strategy to pass information between goals during goal decomposing
  - Design two metrics based on MTree for better expression evaluation

Motivation			
A goal-driven manner	⬅	Seq2Tree	Goal-driven Tree-structured Model (GTS) (Xie and Sun, 2019)
Multi-branch decomposition	⬅	Seq2Exp	Structure-Unified M-Tree Coding Solver (SUMC-Solver) (Bin Wang et al., 2022)
Cross-goal Aattention	⬅	Prev Work	Non-autoregressive Transformer (Gu et al., 2018)
			Pointer Network (Peter Brown et al., 2018)

Get to the Point

Non-autoregressive Math Word Problem Solver with Unified Tree Structure

EMNLP 2023

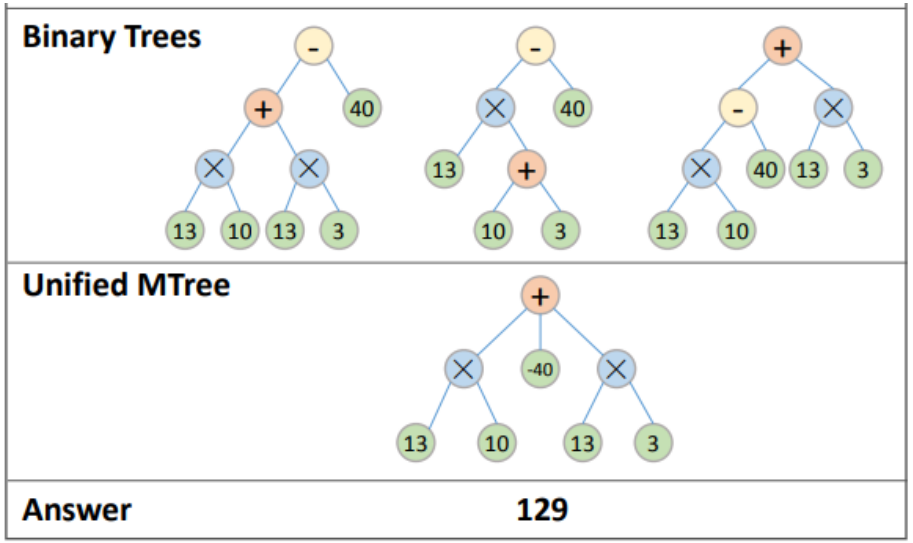
Yi Bin[1], Mengqun Han[2], Wenhao Shi[2], Lei Wang[3], Yang Yang[2]
See-Kiong Ng[1], Heng Tao Shen[2]

[1]National University of Singapore
[2]University of Electronic Science and Technology of China
[3]Singapore Management University

Background

- **The issue of variant expressions**
  - Learn to match exactly the target expression
  - Learn to ignore the alternative ones that can be derived by the arithmetic laws
    - Arithmetic laws: commutative law, associative law, distributive law
  - Can follow either the rules of $income - expense$ or $saturday - sunday$ resulting in different expressions and binary trees

Problem: Dana earn $13 per hour, She worked 10 hours on Saturday and 3 hours on Sunday, and spent $40 on Saturday. How much money did Dana have?	
Reasonable Solutions	$13\times10+13\times3-40$ Sat income + Sun income - all expense $13\times(10+3)-40$ income - expense $(13\times10-40)+13\times3$ Saturday + Sunday



Part 1.

Background

- **The issue of variant expressions**
  - Translate the same problem description to different expression variants is inconsistent with the function mapping
  - Introduce output indeterminacy and make the model difficult to optimize

Result

➡ Problems in expression evaluation due to possible false negatives

➡ Underestimation of the performance of the methods

Address the issue: A unified form of expression

➡ Eliminate the diversity of expression variants and unify them to an identical format with MTree structure (Wang et al., 2022)

➡ Motivation: Goal-driven strategy for tree-structure decoding (Xie and Sun, 2019)
A non-autoregressive MWP solver (MWP-NAS)
to integrate with goal-driven decoding based on MTree

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.
Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. IJCAI. 2019.

Introduction

- **An identical format with MTree structure (Wang et al., 2022)**
  - Contain multiple branches (maybe more than two branches in binary tree)
  - Operator
    - Since the expression diversity is mainly introduced by the computational priority
    - Remove the "−", "/" operators

    ➡ Introduce "+−", "×/" to make all the operators are with the same superiority and the child nodes of the same parent are permutable
  - MTree
    - Represent and learn the mapping between problem and expression
    - Inferior performance: Codes break the relations between numbers (Jie et al., 2022)
    - The code dictionary has to be predefined which limits the flexibility and may incur the issue of out-of-vocabulary (OOV) during inference

    ➡ A non-autoregressive MWP solver: Integrate with goal-driven decoding

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.
Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Introduction

- **A non-autoregressive MWP solver: Integrate with goal-driven decoding**
  - Obtain the semantic representation of problem text via a neural language model
  - Use it as the initial goal for MTree decoding

  - There exist multiple unordered child nodes for each parent
  - ➡ Employ a non-autoregressive Transformer to explore the comprehensive interactions among the nodes and predict them in parallel
  - ➡ For each sub-goal, recursively implement such non-autoregressive decoding until all the child nodes are numbers

  - A cross-goal attention strategy
    - Enable the interactions between goals and make the decomposed sub-goals more accurate

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.

Introduction

- **MTree based evaluation metrics**
  - Each number could be applied in one of four forms $\{\mathrm{n}, \frac{\mathbf{1}}{\boldsymbol{n}}, -\boldsymbol{n}, -\frac{\mathbf{1}}{\boldsymbol{n}}\}$
    - A simple classification module to learn the form
    - The forms is jointly trained with the non-autoregressive decoder
  - Under the current Expression evaluation approaches: **False negative**
    The variants of the same expression would be reported as wrong

➡ MTree based evaluation metrics: **MTree Acc, MTree IoU (Intersection over Union)**

- Evaluate the ability of a solver not only from the final expression
- Consider the partial correctness of the sub-expressions

Introduction

- **MTree based evaluation metrics**
  - MTree Accuracy
    - Such matching accuracy on the whole expression tree fails to measure the partial correctness of expressions
    - But also an important way to evaluate the ability of solvers, as well as humans
  - MTree IoU (Intersection over Union)
    - Calculate the accuracy of paths connecting root and leaves to measure the partial correctness of expressions
    - Transform expressions to Mtrees ➡ construct root-leaf path sets $\boldsymbol{P_p}$ and $\boldsymbol{P_g}$
    - Inspiration from the evaluation in object detection (Ren et al., 2015)

Gold Expression	Predicted Expression	
$13\times10+13\times3-40$	$13\times(10+3)+40$	$(13\times10+3-40)$

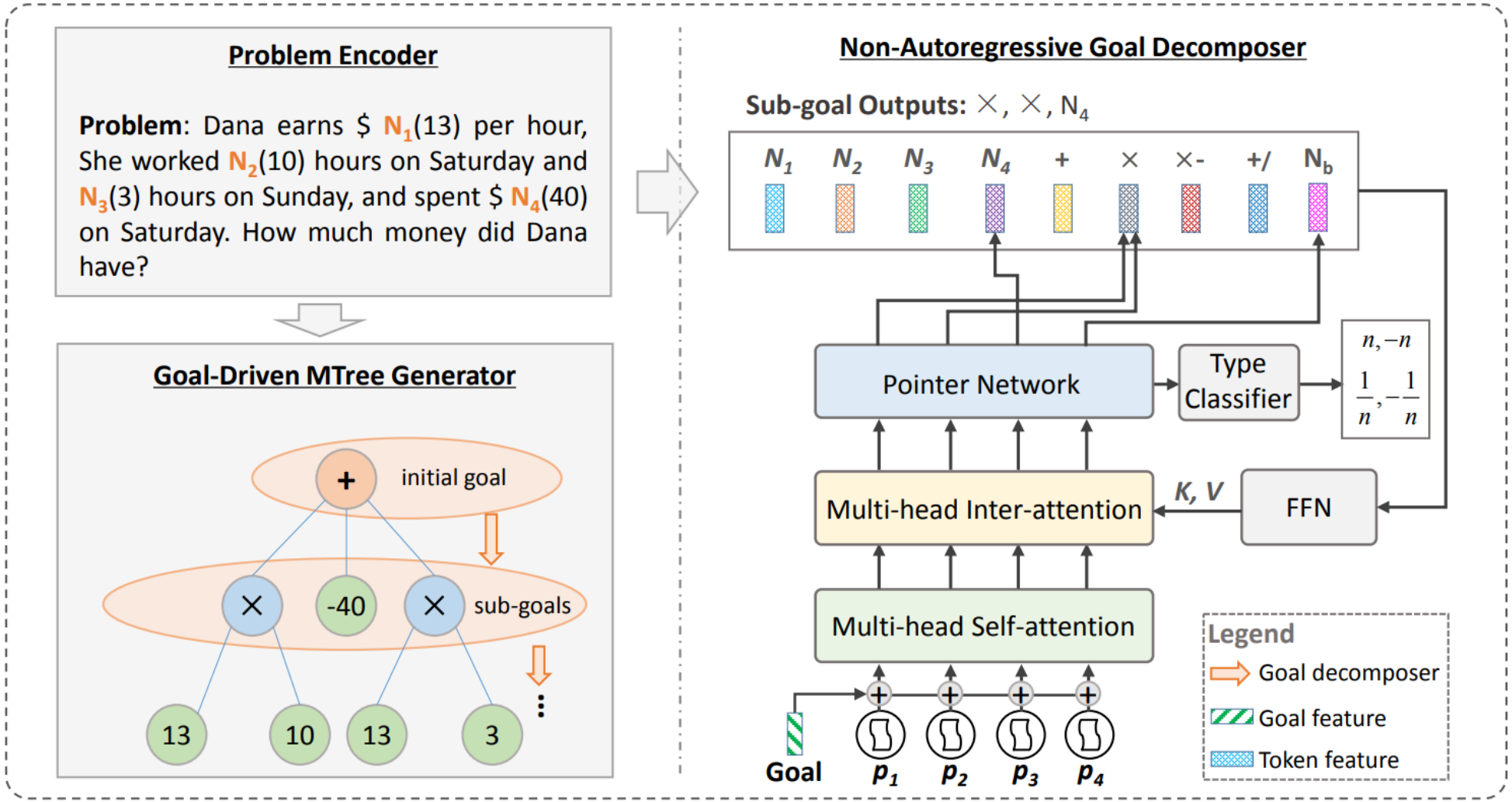
Intersection over Union (IoU)

$$MTreeIoU = \frac{|P_p \cap P_g|}{|P_p \cup P_g|}$$

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Method

- **The overall pipeline of our MWP-NAS**



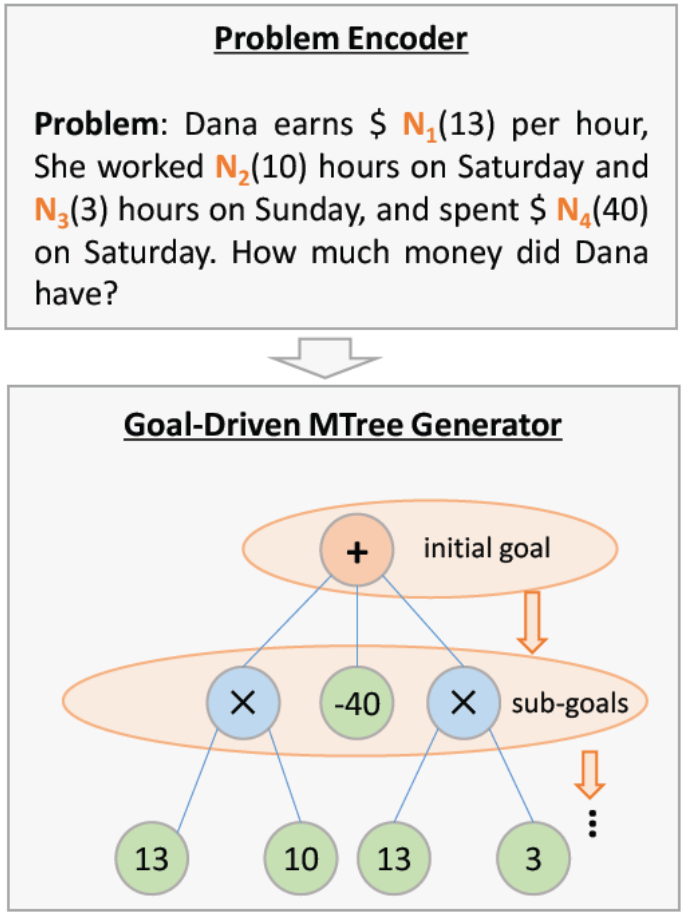
Method

- **Mtree is first introduced to MWP by (Wang et al., 2022)**
  - Transform it to a plain expression by removing the brackets with SymPy1 (Python)
    - The plain expression only contains numbers and four arithmetical operators $\{+, -, \times, /\}$
  - The unswappable operators $\{-, /\}$ can't be applied in multi-branch tree
    - Because different order of operands would lead to different results
  - To tackle this issue, two new operators $\{+-, \times/\}$ are introduced to replace
    - e.g., $\times -$ with operands $\mathbf{\{2, 3\}}$ is equal to $-(\mathbf{2 \times 3})$
    - e.g., $\frac{\mathbf{1}}{\mathbf{2+3}}$ for operands $\mathbf{\{2, 3\}}$
  - Four types of variants $\{\mathbf{n}, \frac{\mathbf{1}}{\boldsymbol{n}}, -\boldsymbol{n}, -\frac{\mathbf{1}}{\boldsymbol{n}}\}$
    - Handle, the different forms of numbers (e.g., negative numbers and fractions)

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.

Method

• Problem Encoder

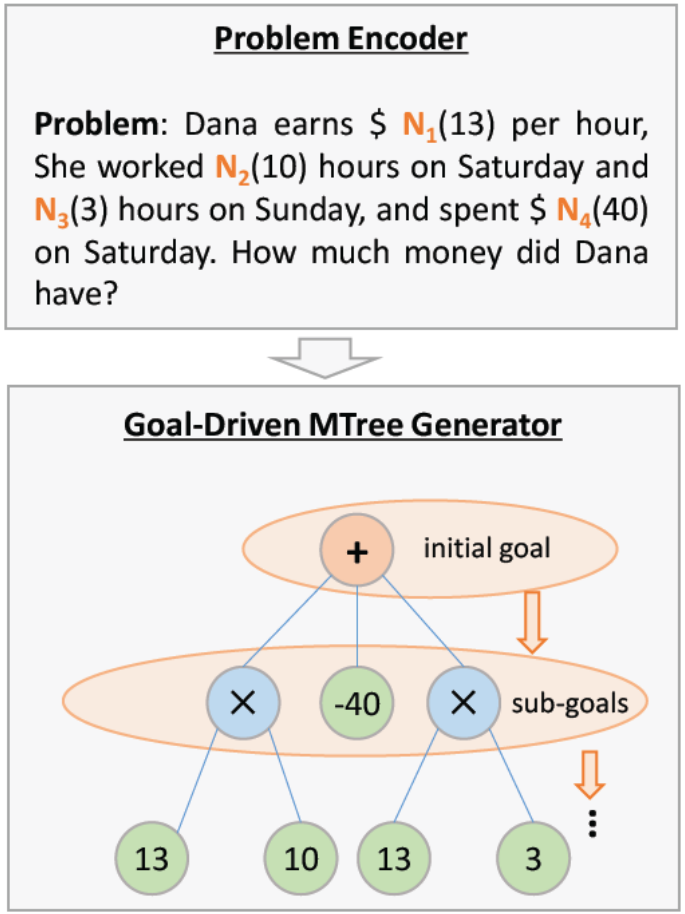


◦ Employ a neural language model to convert the discrete words
- Compact problem representation
- Decode the problem representation to Mtree

◦ Previous works
- The recurrent neural networks (RNN-based) e.g., LSTM or GRU
- The pre-trained language models (PLM-based) e.g., BERT or RoBERTa

Method

• Problem Encoder



◦ Extract the problem representation by PLM

♠ The problem $S = \{w_1, w_2, ..., w_n\}$

♣ Contain numerical values $V = \{v_1, v_2, ..., v_m\}$

- Concatenate a [CLS] and [SEP] at the beginning and end of the problem text, respectively
- The output of [CLS] is employed as the entire problem representation

◦ The problem encoding process

$$E_s, E_V = PLM([CLS]; S; [SEP])$$

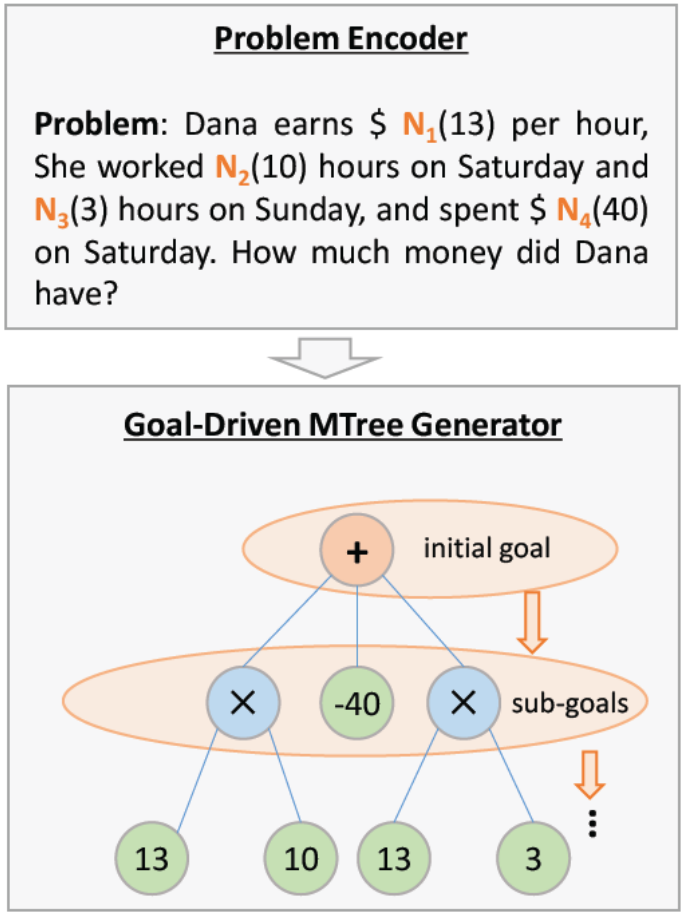
- $\boldsymbol{E_s}$ Output of [CLS] for the entire problem representation
- $\boldsymbol{E_V}$ Contextual representations of numbers

◦ Finetune the PLMs during training

- Make the learned representations better fit the MWP task

Method

• Goal-Driven MTree Generator



◦ Expression tree decoding has been explored in MWP solving
- e.g., sequential decoding with post-order traversal and goal-driven decomposing

➡ MTree generator following the top-down decomposition with a goal-driven mechanism

◦ The root goal $\boldsymbol{E_s}$

◦ Recursively generate the sequence of sub-goals with the top-down fashion, until the sub-goal is not an operand

◦ The sub-goals are categorized as an explicit token
- e.g., operand or operator
- Measure the similarity between sub-goals and candidates

Method

- **Goal-Driven MTree Generator**
  - The candidate representations are defined as

♠ The encoder output E_V ◆ Learned embedding E_*

♣ A special token indicating the end and number of branches of a goal N_b

$$E_c = \begin{cases} E_V, & \text{if it is a number} \\ E_{op}, & \text{if it is an operator} \\ E_{con}, & \text{if it is a constant} \\ E_N, & \text{if it is the special token } N_b \end{cases}$$

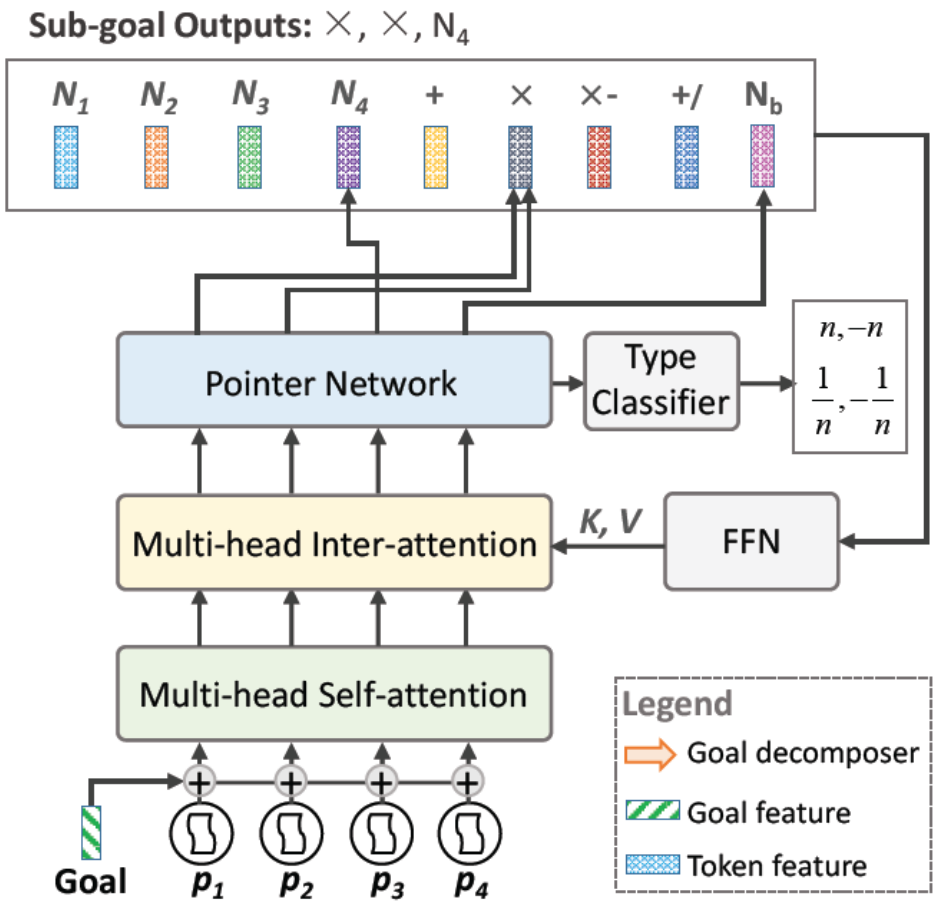
  - The multiple branches of our MTree are unordered (Unlike the binary tree)

  - Different from (Xie and Sun, 2019) generating both sub-goals with different modules
    - Need to treat all the sub-goals the same
    - Generate all the sub-goals with an identical module

Zhipeng Xie, Shichao Sun. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. IJCAI. 2019.

Method

• Non-Autoregressive Goal Decomposer (NAGD)



◦ Handle the unordered multi-branch decomposing as sequence generation

- It is based on the non-autoregressive Transformer (Gu et al., 2018)

◦ The fused representation E_p

$$\tilde{E}_p = \mathrm{MHAtt}(E_p\tilde{W}_Q, E_p\tilde{W}_K, E_p\tilde{W}_V)$$

- Integrate E_g with every positional embedding via element-wise sum
- Embed positions with sin and $\cos$ functions

$$p_{i,2j} = sin(i/10000^{2j/d_k})$$

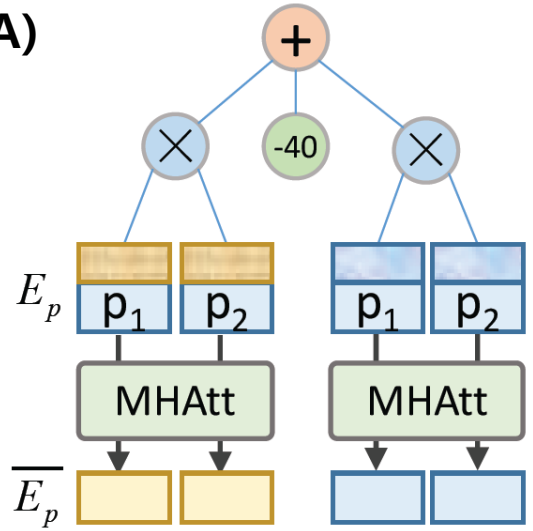
$$p_{i,2j+1} = cos(i/10000^{2j/d_k})$$

J Gu et al. Non-autoregressive neural machine translation. ICLR. 2018.

Method

- **Non-Autoregressive Goal Decomposer (NAGD)**

(A)



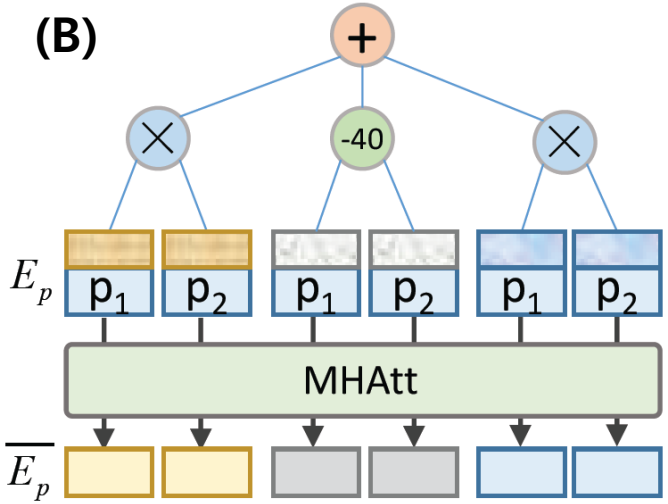
(A) Multi-head self-attention

$$\tilde{E}_p = \mathrm{MHAtt}(E_p\tilde{W}_Q, E_p\tilde{W}_K, E_p\tilde{W}_V)$$

◦ Relative information & dependencies between position

- **When we decompose the left " × " node**
  only capture the information in the left " × " node

(B)



(B) Cross-goal attention

$$\hat{E}_p = softmax(\frac{\tilde{E}_p(E_c\hat{W}_K)^T}{\sqrt{d_k}})(E_c\hat{W}_V)$$

$\boldsymbol{E_p}$ could be used to select the most relevant candidates

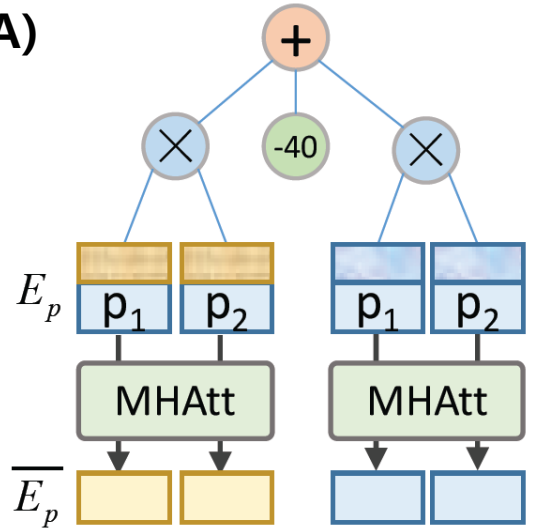
$\boldsymbol{E_c}$ Representation of target candidats

E_V, if it is a number
E_{op}, if it is an operator
E_{con}, if it is a constant
E_N, if it is the special token N_b

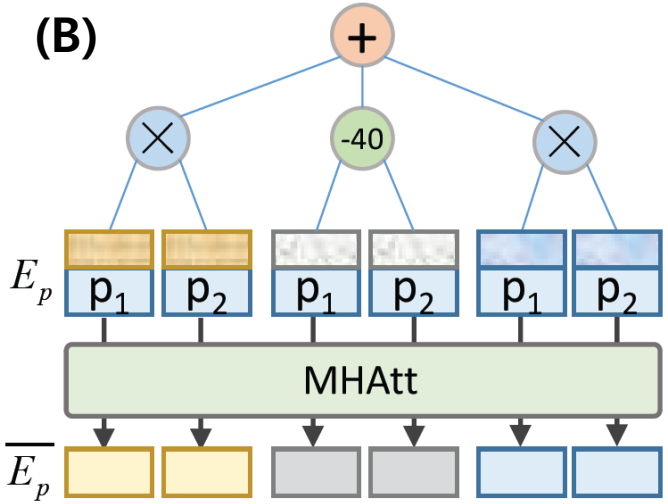
Method

• Non-Autoregressive Goal Decomposer (NAGD)

(A)



(B)



(B) Cross-goal attention

$$\hat{E}_p = softmax(\frac{\tilde{E}_p(E_c\hat{W}_K)^T}{\sqrt{d_k}})(E_c\hat{W}_V)$$

◦ Attach dummy child nodes to pass the information to other child nodes

- **When we decompose the left " × " node**
  peek at the information of nodes " **− 40**" and another " × "
- **For the leaf node " − 40"**
  attach dummy child nodes to pass the information to other child nodes

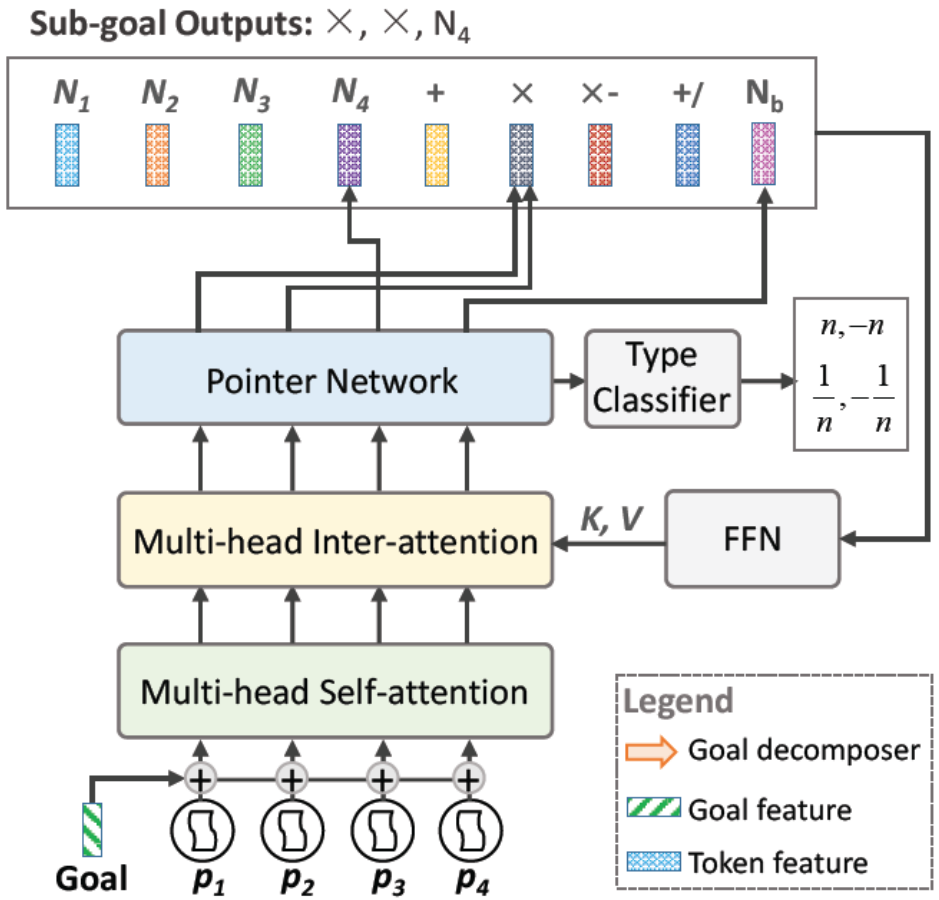
◦ Dummy nodes are used for cross-goal interaction, not for loss computation

$\boldsymbol{E_c}$ Representation of target candidats

E_V,	if it is a number	E_{con},	if it is a constant
E_{op},	if it is an operator	E_N,	if it is the special token N_b

Method

- **Non-Autoregressive Goal Decomposer (NAGD)**



(B) Cross-goal attention

$$\hat{E}_p = softmax(\frac{\tilde{E}_p(E_c\hat{W}_K)^T}{\sqrt{d_k}})(E_c\hat{W}_V)$$

◦ $\boldsymbol{E_P}$ could be used to select the most relevant candidates
  - e.g., operands or operators, via a pointer network (Vinyals et al., 2015)

◦ The probability of position i to choose the $\boldsymbol{j}$-th candidate

$$\omega_{ij} = u^T \tanh(W_p e_i^p + W_b e_j^c)$$

$$Ptr_i = softmax(\omega_i)$$

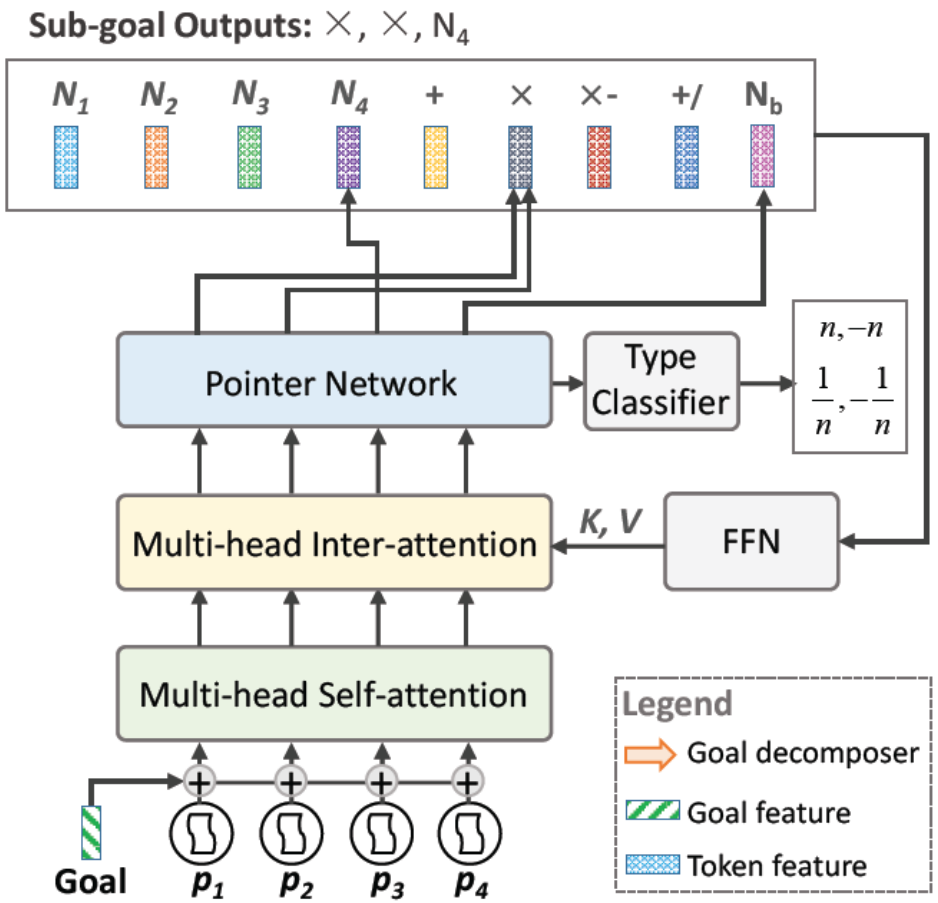
◦ $\boldsymbol{e_i^p}$ and $\boldsymbol{e_j^c}$ The representations of i-th position and candidate

◦ $\boldsymbol{Ptr_i}$ Probabilistic distribution across all the candidates for $\boldsymbol{i}$-th position

J Gu et al. Non-autoregressive neural machine translation. ICLR. 2018.
Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. NeurIPS. 2015.

Method

• Non-Autoregressive Goal Decomposer (NAGD)



◦ To calculate the loss between predictions & ground-truths:
A pseudo order for ground-truth tokens of each goal

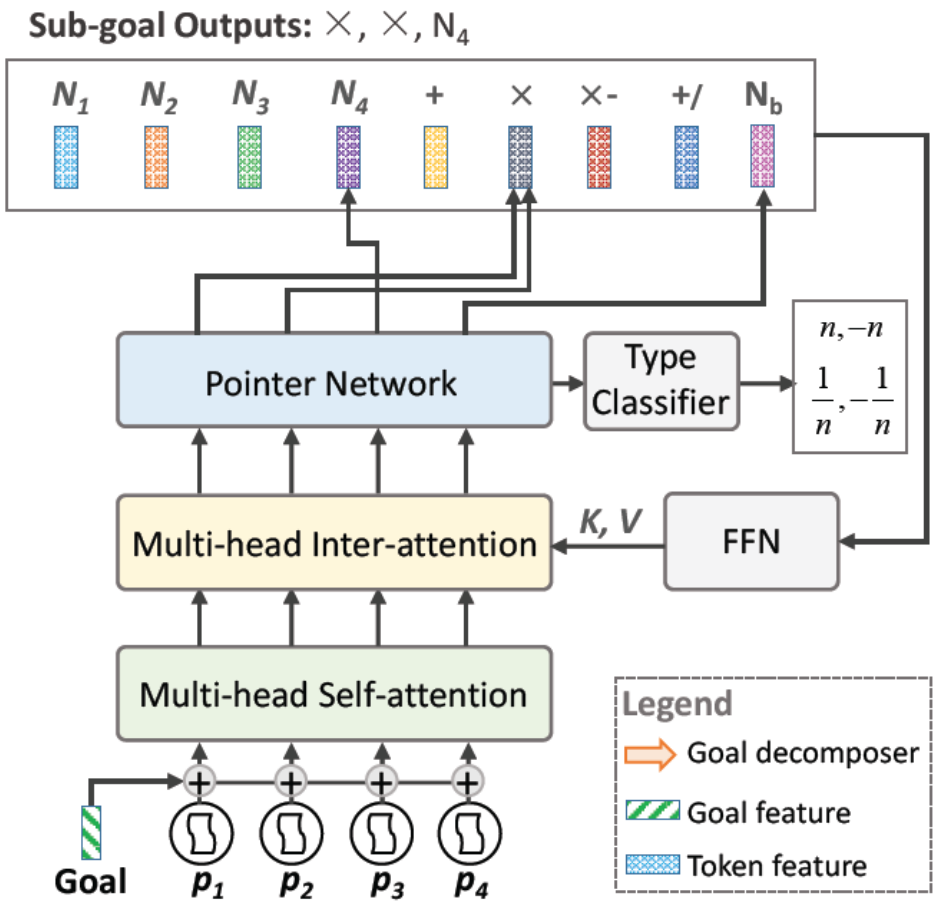
e.g., $[\times, \times, -\mathbf{40}]$ is pseudo order for node " + "

- Align the positions and the ground-truth sub-goals
- The operators are ahead, followed by the constants and operands in the problem

J Gu et al. Non-autoregressive neural machine translation. ICLR. 2018

Method

• Non-Autoregressive Goal Decomposer (NAGD)



◦ Vanilla non-autoregressive Transformer (Gu et al., 2018)
- Fertility prediction to decide the decoding length

◦ Set a max decoding length as $\mathbf{8}$ for all the samples
- More than $\mathbf{99\%}$ samples in the datasets are with MTree branches less than $\mathbf{8}$

◦ Append a special token $\boldsymbol{N_b}$ at the end of sub-goal

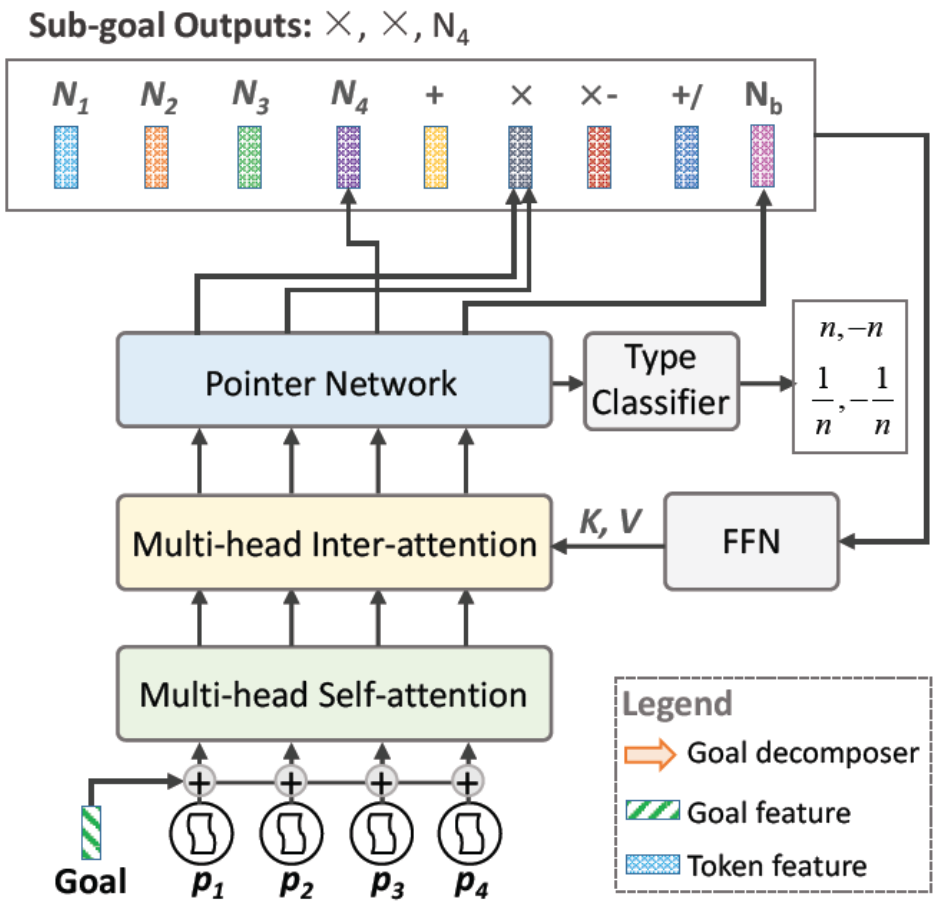
➡ **The number of branches for the current goal decomposing:**

The loss and prediction after the $\boldsymbol{N_b}$ would be ignored during training and inference, respectively

J Gu et al. Non-autoregressive neural machine translation. ICLR. 2018

Method

- **Non-Autoregressive Goal Decomposer (NAGD)**



- After predicting the token of each sub-goal
- Need to indicate the type $\{\mathrm{n}, \frac{1}{n}, -n, -\frac{1}{n}\}$ for predicted operands

Loss Function

- Design a simple MLP for type classification
- Employ focal loss to mitigate the issue of imbalanced classes
- The loss is finally integrated with pointer loss to jointly train the whole MWP-NAS

Evaluation

- **Evaluation Metric**
  - Value accuracy
    - Measures the accuracy of the final answer, but fails to evaluate the validity of the solving procedure
  - Expression accuracy
    - Measure the exact matching accuracy between predicted and ground-truth expression
    - Results in lower performance due to the diverse variants of the same solution expression
  - ➡ The ideal metric
    - The expression accuracy should be the same as value accuracy in theory
    - Handle the reasonable variants of solution expressions
    - All the variants derived from the gold expression with arithmetic rules would lead to the gold answer

Part 4.

Evaluation

- **Evaluation Metric**
  - MTree Accuracy
    - Such matching accuracy on the whole expression tree fails to measure the partial correctness of expressions
    - But also an important way to evaluate the ability of solvers, as well as humans
  - MTree IoU (Intersection over Union)
    - Calculate the accuracy of paths connecting root and leaves to measure the partial correctness of expressions
    - Transform expressions to Mtrees ➡ construct root-leaf path sets $\boldsymbol{P_p}$ and $\boldsymbol{P_g}$
    - Inspiration from the evaluation in object detection (Ren et al., 2015)

Gold Expression
$13\times10+13\times3-40$

Predicted Expression	
$13\times(10+3)+40$	$(13\times10+3-40)$

Intersection over Union (IoU)

$$MTreeIoU = \frac{|P_p \cap P_g|}{|P_p \cup P_g|}$$

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Experiment

- **Datasets**
  - Math23K (Wang et al., 2017)
    - Use the splits **(23162 = 21162 + 1000 + 1000)** (Wang et al., 2022; Zhang et al., 2020)
  - MAWPS (Koncel-Kedziorski et al., 2016)
    - Preprocess for MTree and obtain 2163 samples (Wang et al., 2022)
    - Due to its small size, conduct 5-fold crossvalidation $(\mathbf{2373 = 433 + 431 * 4})$ (Zhang et al., 2020; Jie et al., 2022),
- **Experimental Settings**
  - The maximum size of MTree branch: $\mathbf{8}$
  - Pre-trained language models: **RoBERTa/BERT-base**
  - The hidden size of our non-autoregressive goal decomposer: $\boldsymbol{768\ for\ every\ layer}$
  - $\mathbf{1000}$ and $\mathbf{2000}$ epochs on Math23K and MAWPS for training

Yan Wang et al. Deep neural solver for math word problems. EMNLP. 2017
Rik Koncel-Kedziorski et al. Mawps: A math word problem repository. NAACL-HLT. 2016.

Part 5.

Experiment

• Comparisons with Baselines

Value accuracy (final answer)

Model	Math23K	MAWPS
Seq2Seq (Wang et al., 2017)	58.1	59.5
T-RNN (Wang et al., 2019)	66.9	66.8
GROUP-ATT (Li et al., 2019a)	69.5	76.1
GTS (Xie and Sun, 2019)	75.6	82.6
Graph2Tree (Zhang et al., 2020)	77.4	83.7
NeuralSymbolic (Qin et al., 2021)	75.7	-
HMS (Lin et al., 2021)	76.1	80.3
NUMS2T (Wu et al., 2021)	78.1	-
BERT-Tree (Li et al., 2022)	82.4	-
SAU-Solver (Qin et al., 2020)	76.2♢	75.5♢
UniLM (Dong et al., 2019)	77.5♢	78.0♢
DeductReasoner (Jie et al., 2022)	84.3♠	86.0♠
DeductReasoner (Jie et al., 2022)	85.1♣	91.2♣
SUMC-Solver (Wang et al., 2022)	82.5	82.0
MWP-NAS (SUMC Split)	**84.8**♠	**86.7**♠
MWP-NAS (Reasoner Split)	**86.1**♣	**91.4**♣

- Value accuracy (final answer)
  - Multiple expression variants is hard to evaluate
- If we compare them with expression accuracy
  - It would be many N/A in the comparison
- SUMC Solver employs codes prediction of leaf node to reconstruct the expression tree
  - Code prediction of leaf nodes makes the nodes independent and break relation between the nodes

♢ **The results referred from MTree structure(Wang et al., 2022)**

♠ **The results based on SUMC split**

♣ **5-fold CV train-test setting with data split in DeductReasoner(Jie et al., 2022)**

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.
Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Experiment

- **Effectiveness of Cross-Goal Attention**
  - With such a cross-goal attention mechanism
    - The information belonging to different goals could be passed and aggregated
    - Apparently improves the accuracy of single-goal decomposition
    - Benefit the expressiveness of the model overall

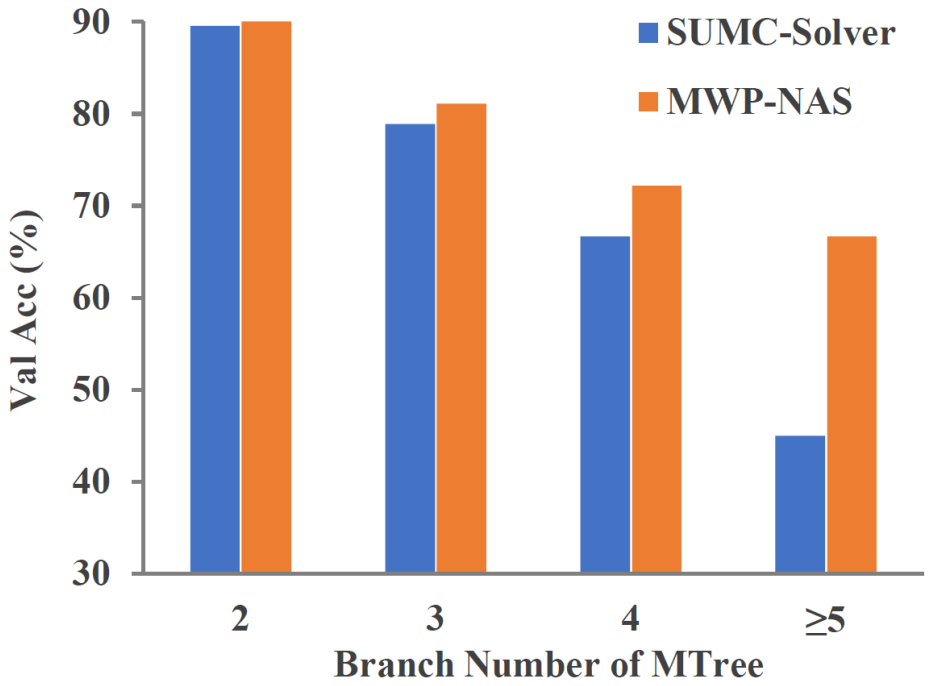
Value accuracy (final answer)

Model	Math23K	MAWPS
w/o Cross-Goal Attention	84.1	86.1
with Cross-Goal Attention	84.8	86.7

Experiment

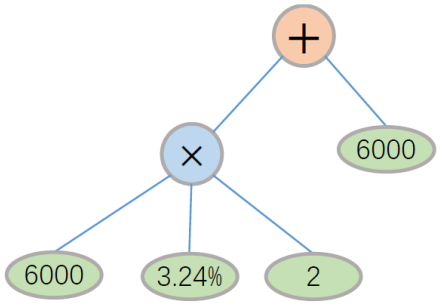
- **Analysis on Different Branch Numbers**
  - MWP-NAS outperforms the SUMC-Solver at all the branch numbers
  - Exhibit more inferior performance for problems with the branch number increasing

Performance comparison on different branch numbers of MTree



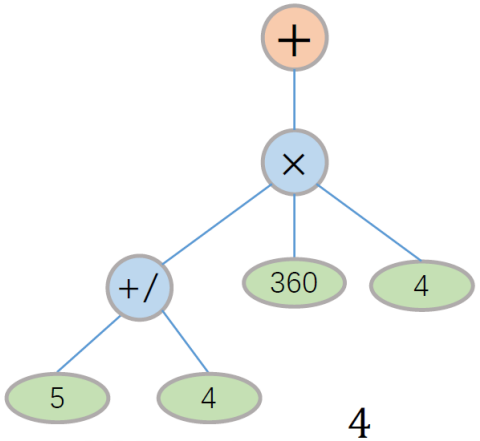
Experiment

- **Illustration with MTree Structure**



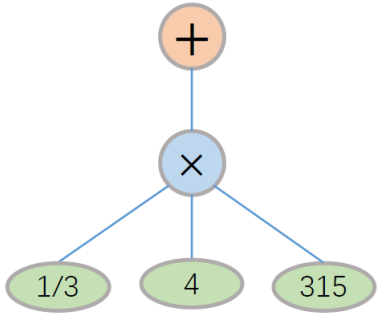
G&P: $6000 + 6000 \times 3.24\% \times 2$

(a)

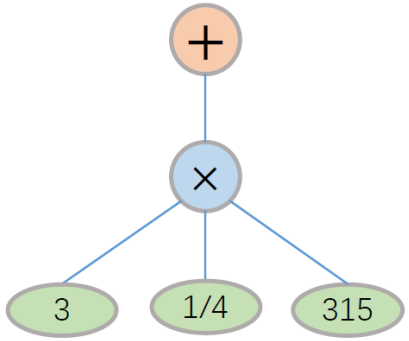


G&P: $360 \times \frac{4}{5+4}$

(b)



G: $315 \times \frac{4}{3}$



P: $315 \times \frac{3}{4}$

(c)

- MWP-NAS is capable of generating accurate MTree structure
- (b) It could easily capture the composition between 5 and 4
- (C) Fail to predict right type for operands, which may because the optimization objectives of number type is an extra loss

➡ Motivate us keep exploration of this direction for further improvements

Part 5.

Experiment

• Evaluation on MTree Metrics

Model	Exp Acc	Val Acc	MTree Acc	MTree IoU
Seq2Seq (Wang et al., 2017)	51.7	58.3	58.2	62.99
GTS (Xie and Sun, 2019)	64.4	75.7	75.3	82.77
Graph2Tree (Zhang et al., 2020)	66.0	77.4	76.9	83.53
DeductReasoner (Jie et al., 2022)	76.4	84.3	83.6	88.86
SUMC-Solver (Wang et al., 2022)	-	82.9	82.4	87.26
MWP-NAS (Ours)	-	84.8	83.8	89.73

- Expression accuracy
  - It should be consistent with value accuracy
  - But it is much lower than value accuracy in practice
- MTree accuracy
  - Handle different variants derived from ground-truth
- MTree IoU
  - Assess the partial correctness of an expression

Problem: The fruit shop sells bananas, oranges and pineapples for a total of 150 kg on Sunday, of which bananas are 27.5 kg, and the number of oranges sold is 3.6 times that of bananas. How many kilograms of pineapples are sold in the fruit shop on Sunday?
Ground-Truth Expression: $150-27.5\times3-27.5$ **DeductReasoner**: $150-27.5-27.5\times3$ **Exp Acc**: ✘ (False) **MTree Acc**: ✔ (True)

Problem: There are 36 chickens, and the number of ducks is twice that of chickens. How many chickens and ducks are there?
Ground-Truth Expression: $(1+2)\times36$ **DeductReasoner**: $36\times2+36$ **Exp Acc**: ✘ (False) **MTree Acc**: ✔ (True)

Bin Wang et al. Structure-unified m-tree coding solver for math word problem. EMNLP. 2022.
Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Experiment

- **Refine the MTree Structure**
  - The plain expression only contains numbers and operators $\{+, -, \times, /\}$
  - To tackle this issue, two new operators $\{+-, \times/\}$ are introduced to replace $\{-, /\}$
    - e.g., $\times -$ with operands $\{2, 3\}$ is equal to $-(2 \times 3)$
    - e.g., $\frac{1}{2+3}$ for operands $\{2, 3\}$
  - Four types of variants $\{n, \frac{1}{n}, -n, -\frac{1}{n}\}$

Results and comparisons with preliminary refined MTree (RefMTree)

Model	Val Acc	MTree Acc	MTree IoU
SUMC-Solver	82.9	82.4	87.26
-RefMTree	79.8	79.5	86.99
MWP-NAS	84.8	83.8	89.73
-RefMTree	84.3	83.4	89.71

All the performances are decreased
Such simple and naive modification for MTree refinement is far from good enough

Part 5.

Experiment

- **Complementary Experimental Results**

Model	MathQA	SVAMP
Seq2Seq (Wang et al., 2017)	-	-
T-RNN (Wang et al., 2019)	-	-
GROUP-ATT (Li et al., 2019a)	70.4♠	21.5♣
GTS (Xie and Sun, 2019)	-	30.8♣
Graph2Tree (Zhang et al., 2020)	69.5♣	**36.5**♣
NeuralSymbolic (Qin et al., 2021)	-	-
HMS (Lin et al., 2021)	-	-
NUMS2T (Wu et al., 2021)	-	-
BERT-Tree (Li et al., 2022)	73.8♣	32.4♣
SAU-Solver (Qin et al., 2020)	-	-
UniLM (Dong et al., 2019)	-	-
DeductReasoner (Jie et al., 2022)	80.6	35.3
SUMC-Solver (Wang et al., 2022)	-	-
MWP-NAS	**81.2**	35.5

♠ **The results referred from Contrastive Learning (Li et al., 2022)**

♣ **The results referred from DeductReasoner (Jie et al., 2022),**

Aida Amini et al. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. NAACL-HLT. 2019.
Arkil Patel et al. Are nlp models really able to solve simple math word problems? NAACL-HLT. 2021.

Conclusion

- **Non-autoregressive MWP solver (MWP-NAS) based on the unified MTree structure**
  - A goal-driven manner for multi-branch decomposition to generate the MTree of expressions
  - A cross-goal attention strategy to pass information between goals during goal decomposing
  - Design two metrics based on MTree for better expression evaluation

Conclusion

- **Limitations**
  - Automatically unifying the expressions could be a small issue
    - Simplify and expand the original expression using Python SymPy package
    - Some cases (about **0.2** percent, i.e., **2** cases in test set) are failed to automatically transform to MTree, due to the expression unification
    - Discard them during training and use the original form as ground-truth for test evaluation
    - Make our reported performance decrease by **0.2** percent
  - With the blooming of large language models (LLMs), we only conducted experiments with some relatively small models

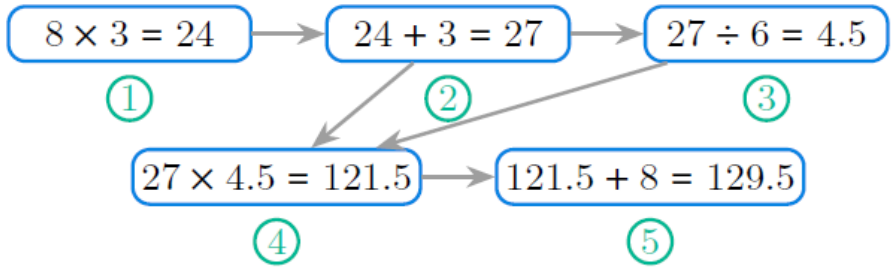
Appendix: DeductReasoner (Jie et al., 2022)

- **Task Definition**
  - Observation: MWP solving can be viewed as a complex relation extraction problem
    - The task of identifying the complex relations among the quantities in the problem text
    - Each primitive arithmetic operation ("+", " − ") defines a different type of relation
  - Learn how to handle the new quantities that emerge from the intermediate expressions
  - Effectively search for the optimal sequence of operations (relations)

A MWP example taken from MathQA

Question: *In a division sum , the remainder is* ***8*** *and the divisor is* ***6*** *times the quotient and is obt--ained by adding* ***3*** *to the thrice of the remainder. What is the dividend?*

Our deductive procedure: 5 ops



Relation extraction between two chosen quantities

- Directly extracts the relation ("×") between 8 and 3
- Context: ("remainder is 8", "thrice of the remainder")
- Allows us to reuse the results from the intermediate expression in the fourth step

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **An approach that explicitly presents deductive reasoning steps**
  - Require voking a relation classification module at each step, yielding a deductive reasonir process
    - Given a problem description $\mathcal{S} = \{w_1, w_2, \cdots, w_n\}$ that consists of a list of n words and $\mathcal{Q}_S = \{q_1, q_2, \cdots, q_m\}$
    - List of m quantities that appear in $\mathcal{S}$, our task is to solve the problem and return the numeric answer
    - Each of the primitive mathematical operations ("+", "−", "×", "÷", "**") above can essentia be used for describing a specific relation between quantities
  - Some questions cannot be answered without relying on certain predefined constants
    - The constants (such as π and 1) may not have appeared in the given problem description
    - Therefore consider a set of constants $\mathcal{C} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$
    - Such constants are regarded as quantities (i.e., $\{q_{m+1}, q_{m+2}, \ldots, q_{m+|\mathcal{C}|}\}$)
    - May play useful roles when forming the final answer expression

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **A Deductive System**
  - Relation (e.g., " + ") between two quantities yields an intermediate expression e
  - At step t, the expression $e^{(t)}$ becomes a newly created candidate quantities
  - One of candidate quantities is ready for deductive reasoning step t+1

Initialization

$$\mathcal{Q}^{(0)} = \mathcal{Q}_{\mathcal{S}} \cup \mathcal{C}$$

At step t

$$e^{(t)}_{i,j,op} = q_i \xrightarrow{op} q_j \quad q_i, q_j \in \mathcal{Q}^{(t-1)}$$

$$\mathcal{Q}^{(t)} = \mathcal{Q}^{(t-1)} \cup \{e^{(t)}_{i,j,op}\}$$

$$q_{|\mathcal{Q}^{(t)}|} := e^{(t)}_{i,j,op}$$

input: q in $\mathcal{Q}^{(0)}$

axiom: $0 : \langle q_1, \cdots, q_{|\mathcal{Q}^{(0)}|} \rangle$

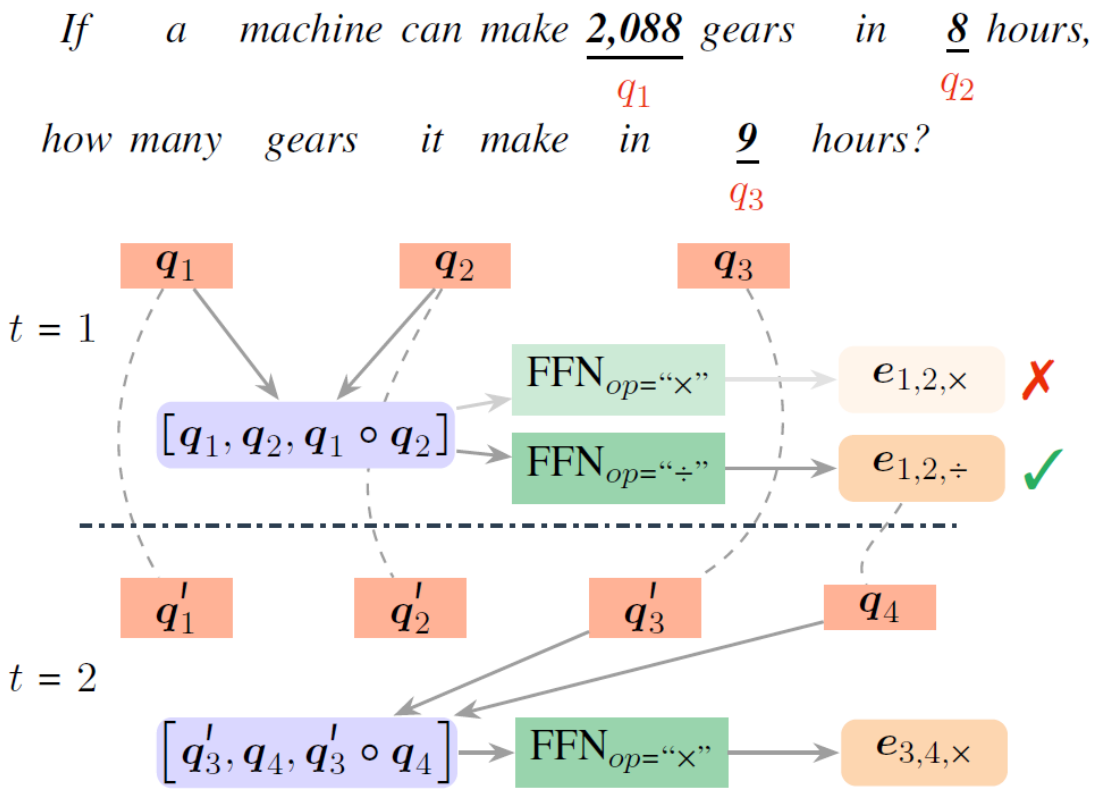
$$q_i \xrightarrow{op} q_j : \frac{t : \langle q_1, \cdots, q_{|\mathcal{Q}^{(t-1)}|} \rangle}{t+1 : \langle q_1, \cdots, q_{|\mathcal{Q}^{(t-1)}|} \mid q_{|\mathcal{Q}^{(t)}|} := e^{(t)}_{i,j,op} \rangle}$$

- $e^{(t)}_{i,j,op}$: The expression after applying the relation op to the ordered pair (q_i, q_j)

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Part 7.

Appendix: DeductReasoner (Jie et al., 2022)



Model architecture for the deductive reasoner

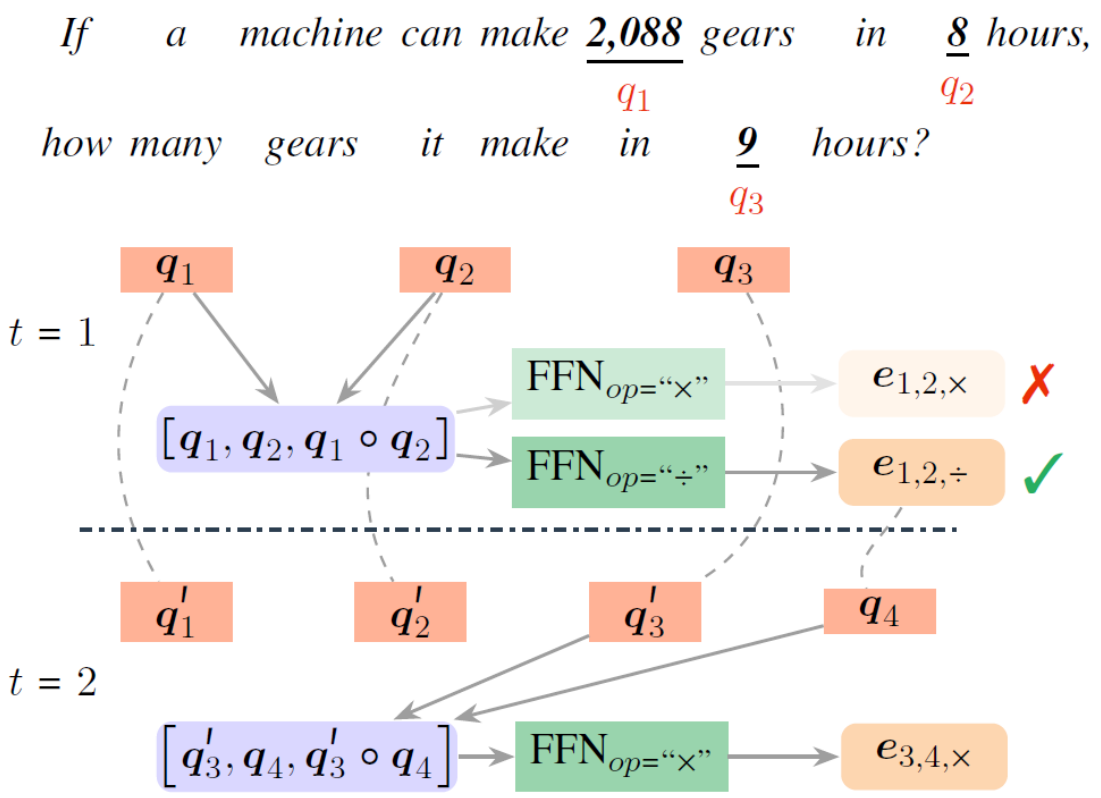
$$\text{“}q_1 \div q_2 \times q_3\text{”}$$

- **Reasoner**
  - Convert the quantities (e.g., 2,088) into a general quantity token " $< quant >$ "
  - Adopt a pre-trained language model such as BERT or ROBERTa
    - Obtain the quantity representation q for each quantity q

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Part 7.

Appendix: DeductReasoner (Jie et al., 2022)



Model architecture for the deductive reasoner

$$\text{“}q_1 \div q_2 \times q_3\text{”}$$

- **Reasoner**
  - Similar to Lee et al. (2017)
  - Obtain the representation of quantity pairs (q_i, q_j)
    - Concatenate the two quantity representations and the element-wise product between them
  - A non-linear feed-forward network (FFN) on top of the pair representation
    - Get representation of newly created expression

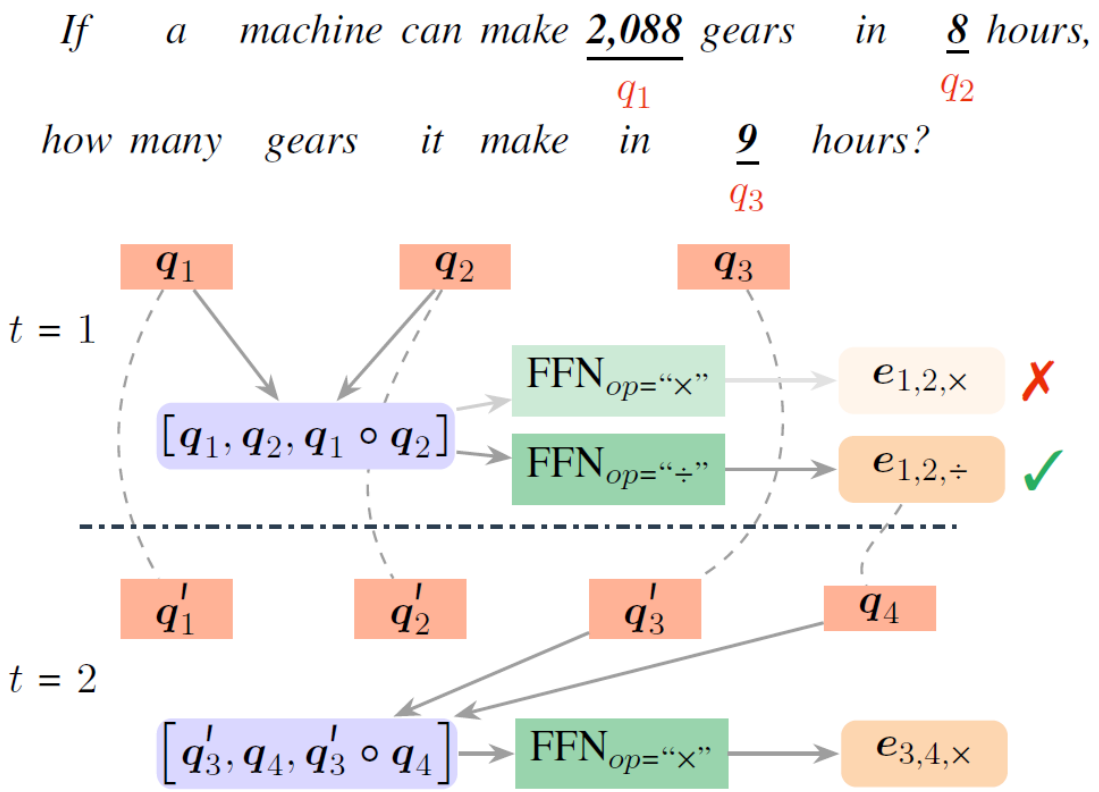
$$\boldsymbol{e}_{i,j,op} = \text{FFN}_{op}([\boldsymbol{q}_i, \boldsymbol{q}_j, \boldsymbol{q}_i \circ \boldsymbol{q}_j]), \quad i \leq j$$

  - $e^{(t)}_{i,j,op}$: The expression after applying the relation op to the ordered pair (q_i, q_j)
  - The constraint $i \leq j$
    - Consider the “reverse operation” ("%","−")
    - The expression $\boldsymbol{e}_{1,2,\div}$ will be regarded as a new quantity with representation $\boldsymbol{q}_4$ at $t = 1$

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Part 7.

Appendix: DeductReasoner (Jie et al., 2022)



Model architecture for the deductive reasoner

$$\text{“}q_1 \div q_2 \times q_3\text{”}$$

- **Reasoner**
  - Assign a score to a single reasoning step that yields the expression $e_{i,j,op}^{(t)}$

$$s(e_{i,j,op}^{(t)}) = s_q(\boldsymbol{q}_i) + s_q(\boldsymbol{q}_j) + s_e(\boldsymbol{e}_{i,j,op})$$

$$s_q(\boldsymbol{q}_i) = \mathbf{w}_q \cdot \text{FFN}(\boldsymbol{q}_i)$$

$$s_e(\boldsymbol{e}_{i,j,op}) = \mathbf{w}_e \cdot \boldsymbol{e}_{i,j,op}$$

  - Find the optimal expression sequence

$$[e^{(1)}, e^{(2)}, \cdots, e^{(T)}]$$

    - Enables us to compute the final numerical answer
    - T The total number of steps required for this deductive process

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Terminator**
  - A mechanism that decides whether the deductive procedure is ready to terminate at any given time
  - A binary label τ : 1 The procedure stops here, 0 otherwise

The final score of the expression *e* at time step *t*

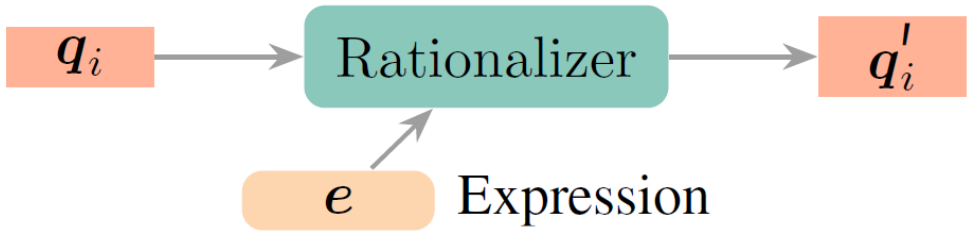
$$S(e^{(t)}_{i,j,op}, \tau) = s(e^{(t)}_{i,j,op}) + \mathbf{w}_\tau \cdot \mathrm{FFN}(\boldsymbol{e}_{i,j,op})$$

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Rationalizer**
  - Rationalization
    - Potentially give us the rationale that explains an outcome
    - Obtain a new intermediate expression at step t, it is crucial to update the representations for the existing quantities
  - If the quantity representations don't get updated with the deductive reasoning
    - Those expressions that were initially highly ranked (say, at the first step) would always be preferred over those lowly ranked ones throughout the process

Rationalizing quantity representation



- The intermediate expression $\boldsymbol{e}$ serves as the rationale that explains how the quantity changes from q to q'

$$\boldsymbol{q}_i' = \text{Rationalizer}(\boldsymbol{q}_i, \boldsymbol{e}^{(t)}) \quad \forall\ 1 \le i \le |\mathcal{Q}|$$

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Rationalizer**
  - Adopt well-known techniques as rationalizers
    - Allow us to update the quantity representation with the intermediate expression representation
  - Multi-head self-attention (Vaswani et al., 2017)
    - Construct a sentence with token representations (Quantity q_i & Previous expression e)
  - A gated recurrent unit (GRU) (Cho et al., 2014) cell
    - Use q_i as the input state and e as the previous hidden state in a GRU cell

The mechanism in different rationalizers

Rationalizer	Mechanism
Multi-head Self-Attention	$\text{Attention}(Q = [\boldsymbol{q}_i, \boldsymbol{e}], K = [\boldsymbol{q}_i, \boldsymbol{e}], V = [\boldsymbol{q}_i, \boldsymbol{e}])$
GRU cell	$\text{GRU\_Cell}(\text{input} = \boldsymbol{q}_i, \text{previous hidden} = \boldsymbol{e})$

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Training**
  - Adopt the teacher-forcing strategy (Williams and Zipser, 1989)
    - Similar to training sequence-to-sequence models (Luong et al., 2015)
    - Guide the model with gold expressions during training

Loss Function

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \Big(\max_{(i,j,op)\in\mathcal{H}^{(t)},\tau} \Big[\mathcal{S}_{\boldsymbol{\theta}}(e^{(t)}_{i,j,op}, \tau) \Big] - \mathcal{S}_{\boldsymbol{\theta}}(e^{(t)}_{i^*,j^*,op^*}, \tau^*) \Big) + \lambda ||\boldsymbol{\theta}||^2$$

$\boldsymbol{\theta}$ All parameters in the deductive reasoner

$\mathcal{H}^{(t)}$ All the possible choices of quantity pairs and relations available at time step t

λ The hyperparameter for the L_2 regularization term

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Inference**
  - Set a maximum time step T_{max} and find the best expression e^* that has highest score at each time step
  - Once we see $\tau = 1$ is chosen, we stop constructing new expressions and terminate the process
  - The overall expression
    - It will formed by the resulting expression sequence
    - It will be used for computing the final numerical answer

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Declarative Constraints**
  - Model repeatedly relies on existing quantities to construct new quantities
  - Model results in a structure showing the deductive reasoning process
  - It allows certain declarative knowledge to be conveniently incorporated

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \Big(\max_{(i,j,op)\in\mathcal{H}^{(t)},\tau} \Big[\mathcal{S}_{\boldsymbol{\theta}}(e^{(t)}_{i,j,op}, \tau) \Big] - \mathcal{S}_{\boldsymbol{\theta}}(e^{(t)}_{i^*,j^*,op^*}, \tau^*) \Big) + \lambda ||\boldsymbol{\theta}||^2$$

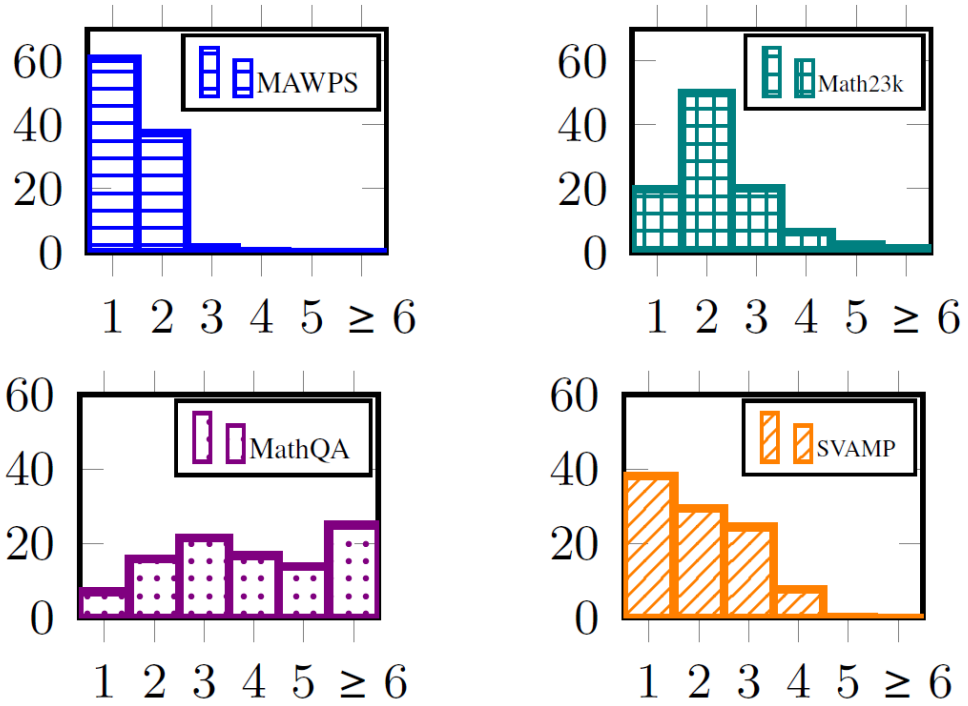
The default approach considers all the possible combinations among the quantities during the maximization step

  - Easily impose constraints to avoid considering certain combinations
    - Find in certain datasets such as SVAMP, there does not exist any expression that involve operations applied to the same quantity ($9+9$, 9×9)
  - Observe that the intermediate results would not be negative.
    - Simply exclude such cases in the maximization process, effectively reducing the search space during both training and inference

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: DeductReasoner (Jie et al., 2022)

- **Datasets**
  - MAWPS (Koncel-Kedziorski et al., 2016), SVAMP (Patel et al., 2021) ("+", "−", "×", "÷")
  - Math23k (Wang et al., 2017), MathQA (Amini et al., 2019) ("+", "−", "×", "÷", "**")
    - MathQA: Follow Tan et al. (2021) to adapt the dataset to filter out some questions that are unsolvable



Percentage of questions with different operation count

- MAWPS
  - 97% can be answered with only one or two operations
- MathQA
  - More than 60% have three or more operations
  - GRE questions in many domains including physics, geometry, probability, etc
- SVAMP
  - Variations from MAWPS: adding extra quantities, swapping the positions between noun phrases, etc.

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Part 7.

Appendix: DeductReasoner (Jie et al., 2022)

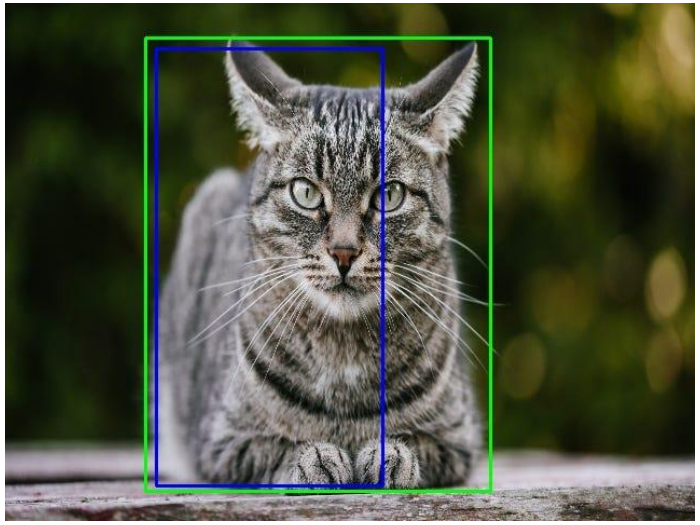
- **Results**

#Operation	MAWPS		Math23k		MathQA		SVAMP	
	Baseline	OURS	Baseline	OURS	Baseline	OURS	Baseline	OURS
1	88.2	**92.7**	91.3	**93.6**	**77.3**	**77.4**	**51.9**	**52.0**
2	91.3	**91.6**	89.3	**92.0**	81.3	**83.5**	17.8	**32.1**
3	-	-	74.5	**77.0**	81.9	**83.4**	-	-
4	-	-	59.1	**60.3**	79.3	**81.7**	-	-
>=5	-	-	56.5	**69.2**	**71.5**	**71.4**	-	-
Overall Performance								
Equ Acc.	80.8	88.6	71.2	79.0	74.0	74.0	40.9	45.0
Val Acc.	88.7	92.0	82.4	85.1	77.1	78.6	43.8	47.3

Zhanming Jie et al. Learning to reason deductively: Math word problem solving as complex relation extraction. ACL. 2022.

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

- **IOU(Intersection over Union)**
  - A term used to describe the extent of overlap of two boxes
  - The greater the region of overlap, the greater the IOU
  - Applications related to object detection
    - Train a model to output a box that fits perfectly around an object
  - It is used in non max suppression
    - It is used to eliminate multiple boxes that surround the same object, based on which box has a higher confidence



Depiction of the task of object detection

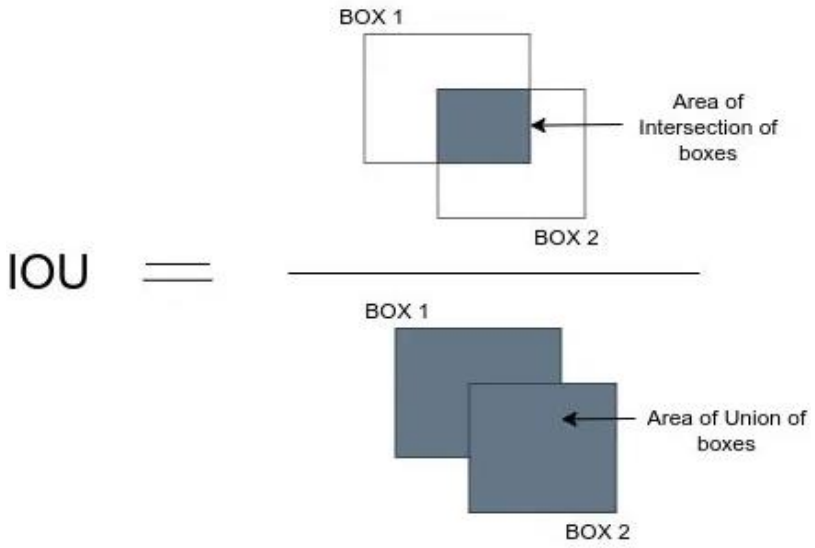
- The green box represents the correct box
- The blue box represents the prediction from our model
- The aim of this model would be to keep improving its prediction
  - i.e the IOU between the two boxes becomes equal to 1

Vineeth S Subramanyam, https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef

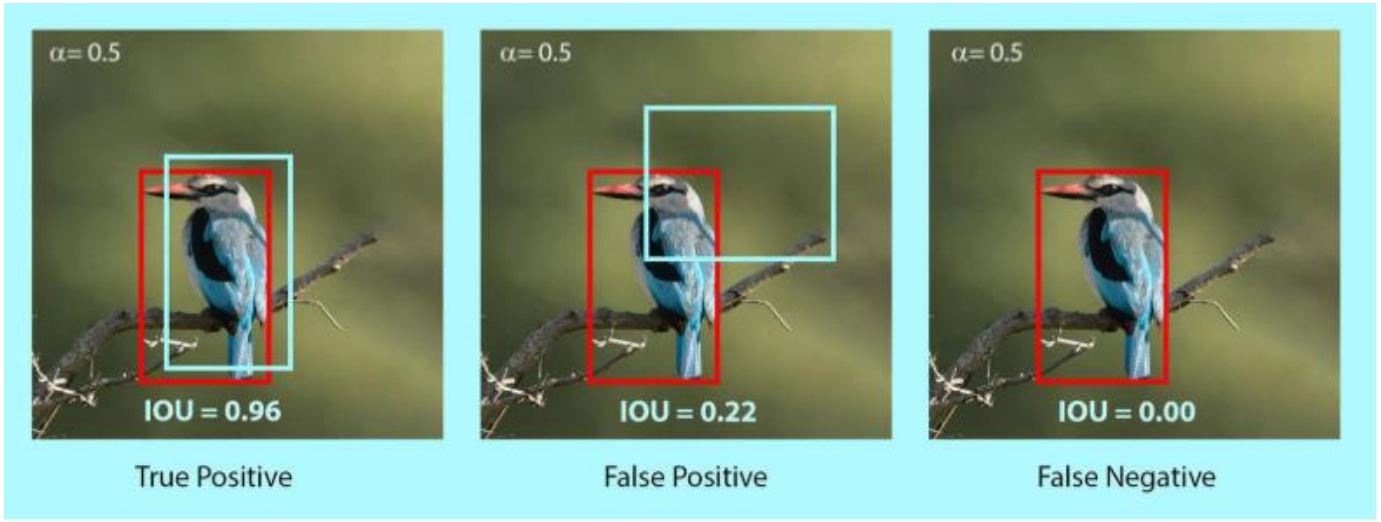
Appendix: Intersection over Union (IoU) (Ren et al., 2015)

- **IOU(Intersection over Union)**
  - $IOU = \frac{Area\ of\ Intersection\ of\ boxes}{Area\ of\ Union\ of\ boxes} = \frac{Area\ of\ Intersection\ of\ boxes}{Area\ of\ box\ 1 + Area\ of\ box\ 2 - Area\ of\ Intersection\ of\ boxes}$
  - With the help of the IoU threshold, we can decide whether the prediction is True Positive(TP), False Positive(FP), or False Negative(FN)

The formula of IOU



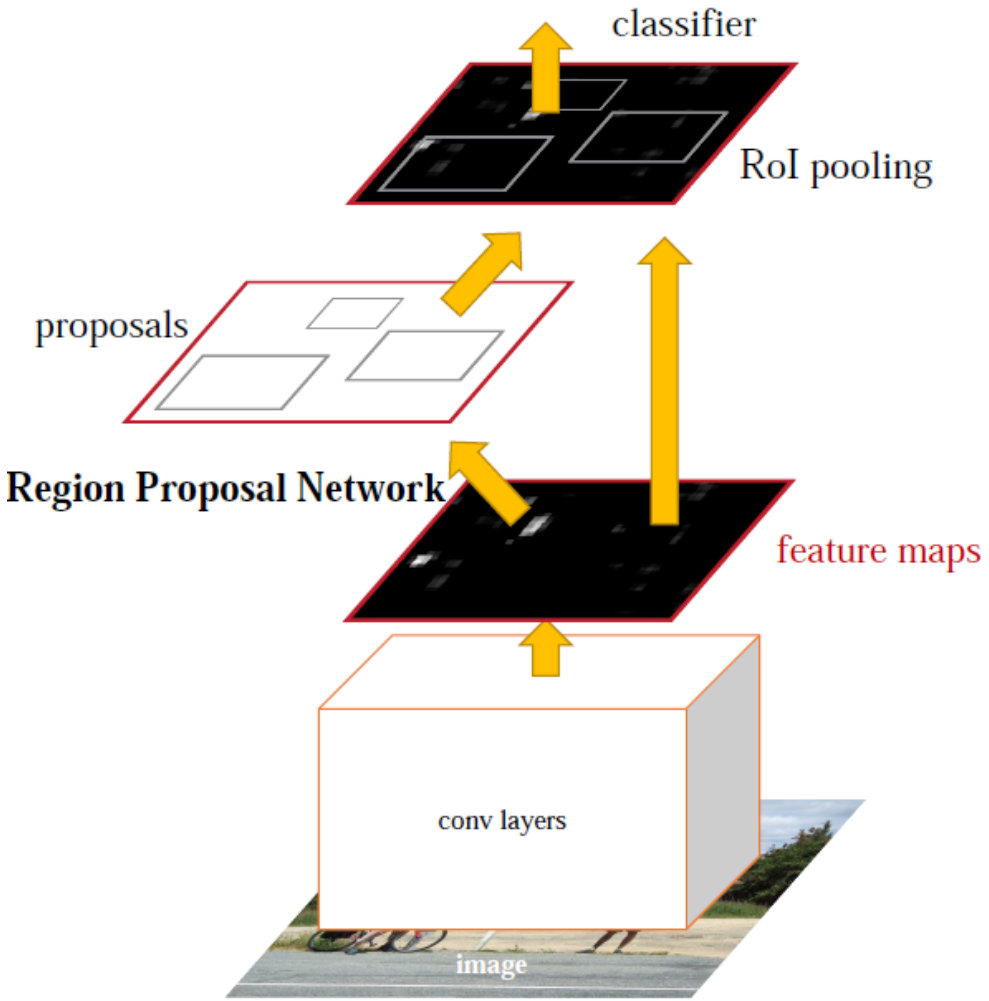
Intersection of two boxes



Vineeth S Subramanyam, https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

A single, unified network for object detection



- **FASTER R-CNN**
  - The first module
    - A deep fully convolutional network that proposes regions
  - The second module
    - The Fast R-CNN detector that uses the proposed regions
  - The entire system is a single, unified network for object detection
  - Using the recently popular terminology of neural networks with 'attention' mechanisms, the RPN module tells the Fast R-CNN module where to look.

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

- **Anchors**
  - At each sliding-window location, we simultaneously predict multiple region proposals
    - The number of maximum possible proposals for each location is denoted as k
  - The reg layer has $4k$ outputs encoding the coordinates of k boxes
  - The cls layer outputs $2k$ scores that estimate probability of object or not object for each proposal
  - The k proposals are parameterized relative to k reference boxes, which we call anchors
  - An anchor is centered at the sliding window in question
  - An anchor is associated with a scale and aspect ratio
  - Use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

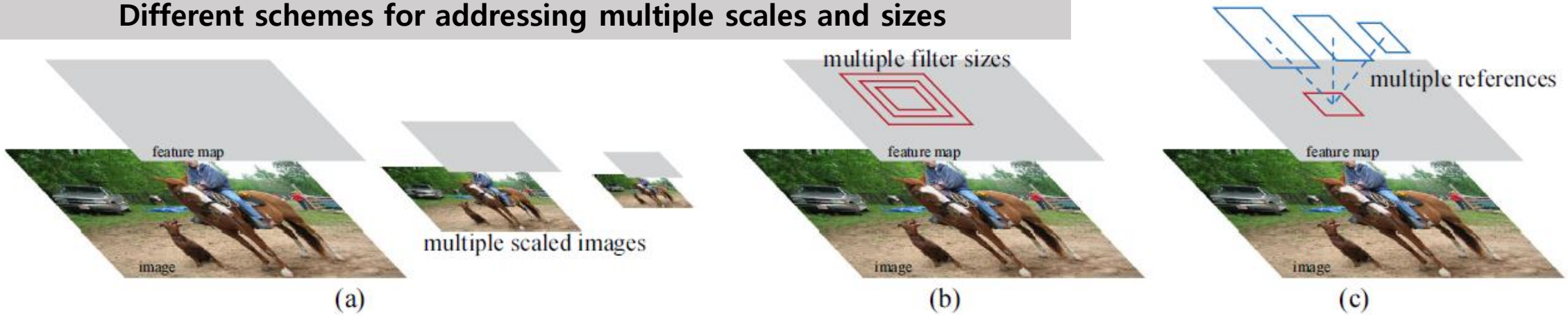
- **Translation-Invariant Anchors**
  - An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors
  - The translation-invariant property also reduces the model size

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

- **Multi-Scale Anchors as Regression References**
  - Our design of anchors presents a novel scheme for addressing multiple scales size
  - As a comparison, our anchor-based method is built on a pyramid of anchors, which is more cost-efficient
  - Our method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios

Different schemes for addressing multiple scales and sizes



- (a) Pyramids of images and feature maps are built, and the classifier is run at all scales
- (b) Pyramids of filters with multiple scales/sizes are run on the feature map
- (c) We use pyramids of reference boxes in the regression functions

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Appendix: Intersection over Union (IoU) (Ren et al., 2015)

- **Loss Function**
  - For training RPNs, we assign a binary class label to each anchor
    - (i) The anchor/anchors with the highest Intersection-over- Union (IoU) overlap with a ground-truth box
    - (ii) An anchor that has an IoU overlap higher than 0.7 with any ground-truth box
  - A single ground-truth box may assign positive labels to multiple anchors
    - Adopt the first condition for the reason that in some rare cases the second condition may find no positive sample
  - Assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes
    - Anchors that are neither positive nor negative do not contribute to the training objective

The multi-task loss in Fast R-CNN

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Shaoqing Ren et al. Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS. 2015.

Appendix

- **Math23K (Wang et al., 2017)**
  - 23,161 problems labeled with structured equations and answers
  - Crawl over 60,000 Chinese math word problems from a couple of education web sites
  - Real math word problems for elementary school students
  - A rule-based extraction method
    - Extract equation systems and structured answers from the solution text
    - Achieves very high precision and medium reca

Problem: Dan have 5 pens and 3 pencils, Jessica have 4 more pens and 2 less pencils than him. How many pens and pencils do Jessica have in total?
Equation: x = 5 + 4 +3 -2
Solution: 10

Yan Wang et al. Deep neural solver for math word problems. EMNLP. 2017

Problem Formulation

- A problem P can be solved by a mathematical equation E_p formed by V_p and mathematical operators
- To decrease the diversity of equations
  - Map each equation to an equation template T_p through a number mapping M_p

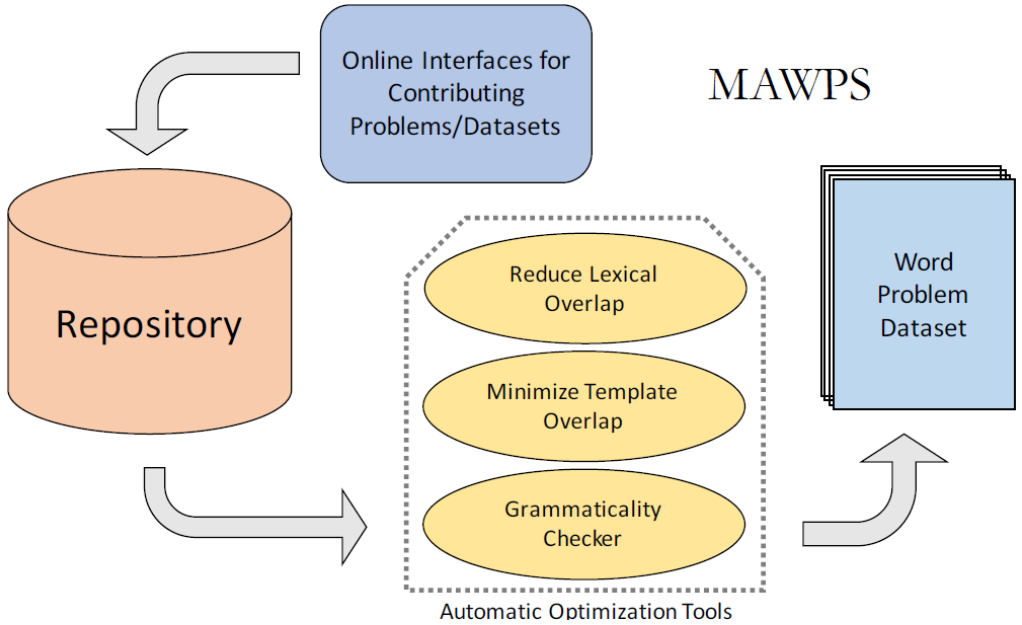
M_p $M : \{n_1 = 5; \quad n_2 = 3; \quad n_3 = 4; \quad n_4 = 2;\}$

T_p $x = n_1 + n_3 + n_2 - n_4$

Appendix

- **MAWPS (Koncel-Kedziorski et al., 2016)**
  - Compile a dataset of math word problems of varying complexity from different websites
  - Design an algorithm that addresses different problem types while still being robust to the template or lexical variations has remained a challenge

MAWPS System Overview



Training Dataset Example

Question	Equation	Answer	Numbers
There school 's baseball team had N_00 new players and N_01 returning players . If the coach put them into groups with N_02 players in each group , how many groups would there be ?	(N_00 + N_01) / N_02	2	2.0 6.0 4.0

Rik Koncel-Kedziorski et al. Mawps: A math word problem repository. NAACL-HLT. 2016

Part 7.

Appendix

- **MAWPS (Koncel-Kedziorski et al., 2016)**
  - Reduce Lexical Overlap
    - At each step, add a problem which has the least average pairwise lexical overlap with the problems already in D'
  - Minimize Template Overlap
    - If there are no new templates to be added, then a p whose template is least frequent among those of D' is selected
  - Grammaticality
    - Filter problems with the broad-coverage, linguistically precise English Resource Grammar

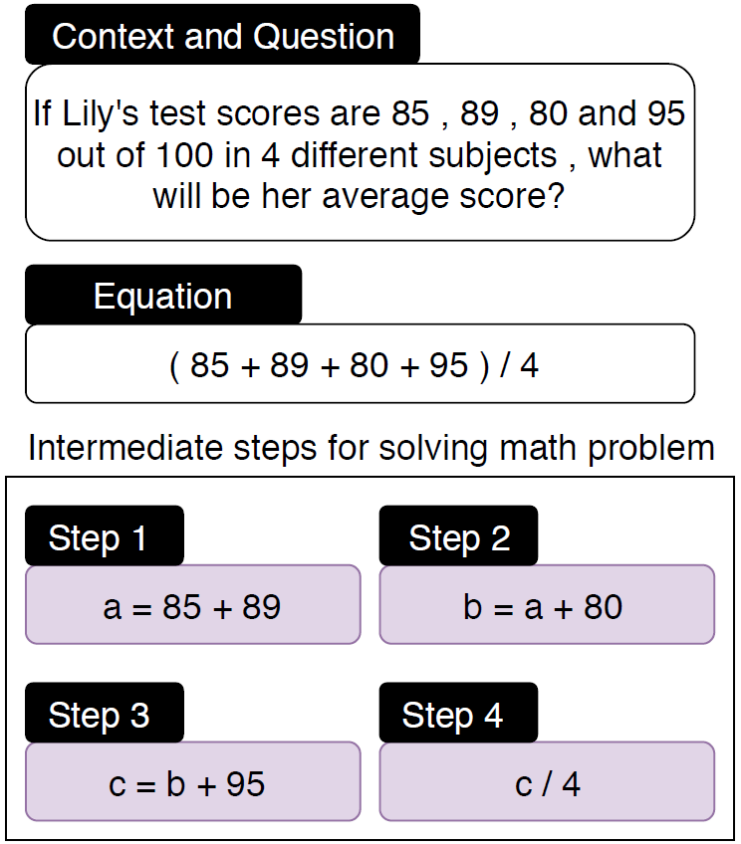
Characteristics of publicly available datasets added to our repository

Dataset	# Probs$\|D\|$	# Gramm.	Lexical Overlap (Lex)			Template Overlap (Tmpl)		
			$k = \|D\|/2$	$k = \|D\|$	Reduction	$k = \|D\|/2$	$k = \|D\|$	Reduction
AddSub	395	357	6.1	7.9	22.8	33	37.2	11.3
SingleOp	562	491	6.1	7.8	21.8	24.7	25.4	2.8
MultiArith	600	526	7.8	9.4	17.0	19.7	22.1	10.9
SingleEq	508	434	5.4	6.8	20.6	11	17.9	38.5
SimulEq-S	514	437	4.7	6	21.7	2.9	12.5	76.8
SimulEq-L	1155	980	4.4	5.7	22.8	0.1	3.3	97.0

Rik Koncel-Kedziorski et al. Mawps: A math word problem repository. NAACL-HLT. 2016

Appendix

- **MathQA (Amini et al., 2019)**
  - A new large-scale, diverse dataset of 37k English multiple-choice math word problems
  - Require logical reasoning about implied actions and relations between entities



Example of a math word problem

- The complexity of the problem-solving task
  - Deduce implied constants (pi) and knowledge of domain-specific formulas (area of the square)
- Select implied operations and arguments
- Generate a program of intermediate steps for solving a math word problem
- Operations can be dependant to the previous ones by the values they use as arguments

The general form

$$o_1(\mathbf{a}_1)o_2(\mathbf{a}_2)...o_n(\mathbf{a}_n)$$

Operation Sequence generated by model

$$\text{add}_1(85, 89)\text{add}_2(174, 80)$$
$$\text{add}_3(254, 95)\text{divide}_4(349, 4)$$

Aida Amini et al. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. NAACL-HLT. 2019

Part 7.

Appendix

- **SVAMP (Patel et al., 2021)**
  - The majority of problems in benchmark datasets
    - It can be solved by shallow heuristics lacking word-order information or lacking question text
  - Grade level up to 4 by applying simple variations over an existing dataset
    - Apply certain types of variations to a set of seed examples sampled from the ASDiv-A dataset
  - Highlights the brittle nature of existing models when trained on these benchmark

Example of a Math Word Problem along with the types of variations that we make to create SVAMP

PROBLEM:
Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Jack have now?
Equation: 8 - 3 = 5

QUESTION SENSITIVITY VARIATION:
Text: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Mary have now?
Equation: 5 + 3 = 8

REASONING ABILITY VARIATION:
Text: Jack had 8 pens and Mary had 5 pens. Mary gave 3 pens to Jack. How many pens does Jack have now?
Equation: 8 + 3 = 11

STRUCTURAL INVARIANCE VARIATION:
Text: Jack gave 3 pens to Mary. If Jack had 8 pens and Mary had 5 pens initially, how many pens does Jack have now?
Equation: 8 - 3 = 5

Arkil Patel et al. Are nlp models really able to solve simple math word problems? NAACL-HLT. 2021

Part 7.

Appendix

- **SVAMP (Patel et al., 2021)**
  - Question Sensitivity
    - Same Object, Different Structure
    - Different Object, Same Structure
    - Different Object, Different Structure
  - Reasoning Ability
    - Add relevant information
    - Change Information
    - Invert Operation
  - Structural Invariance
    - Change order of objects
    - Change order of phrases
    - Add irrelevant information

Change in accuracies when categories are removed

Removed Category	# Removed Examples	Change in Accuracy (Δ)
Question Sensitivity	462	+13.7
Reasoning Ability	649	-3.3
Structural Invariance	467	+4.5

Change in accuracies when variations are removed

Removed Variation	# Removed Examples	Change in Accuracy (Δ)
Same Obj, Diff Struct	325	+7.3
Diff Obj, Same Struct	69	+1.5
Diff Obj, Diff Struct	74	+1.3
Add Rel Info	264	+5.5
Change Info	149	+3.2
Invert Operation	255	-10.2
Change order of Obj	107	+2.3
Change order of Phrases	152	-3.3
Add Irrel Info	281	+6.9

Arkil Patel et al. Are nlp models really able to solve simple math word problems? NAACL-HLT. 2021