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

Part 1.

Background

- **Math word problem solver**
  - Need to output an expression for solving the unknown variable asked in the problem
    - Problem that consists of mathematical operands (numerical values) and operation symbols

$$(+, -, \times, \div)$$

  - Good testbeds for evaluating the intelligence level of agents (Lin et al., 2021)
    - Solver understands the natural-language problem
    - Solver is able to model the relationships between the numerical values to perform arithmetic reasoning

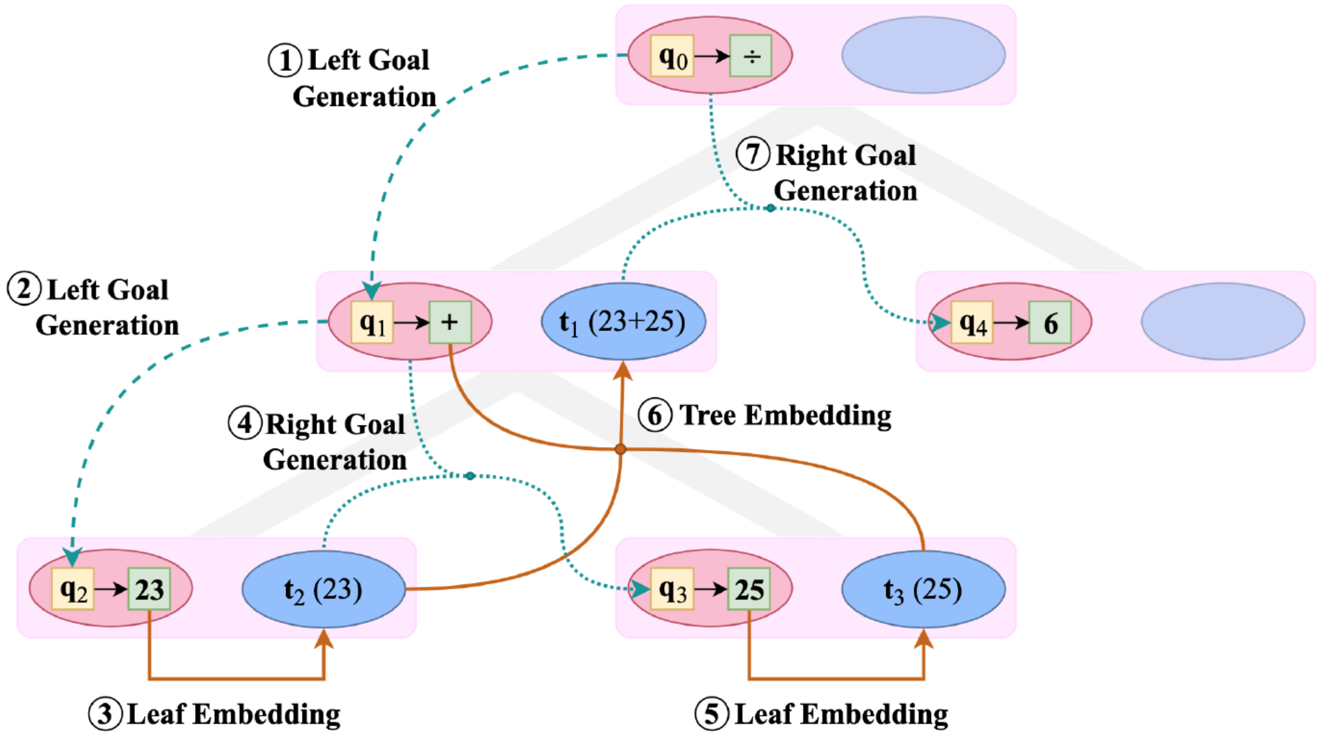
Problem: Dana earn $13 per hour, She worked 10 hours on Saturday and 3 hours on Sunday, and spent $40 on Saturday. How much money did Dana have?

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

Xin Lin et al. HMS: A hierarchical solver with dependency-enhanced understanding for math word problem. AAAI. 2021.

Background

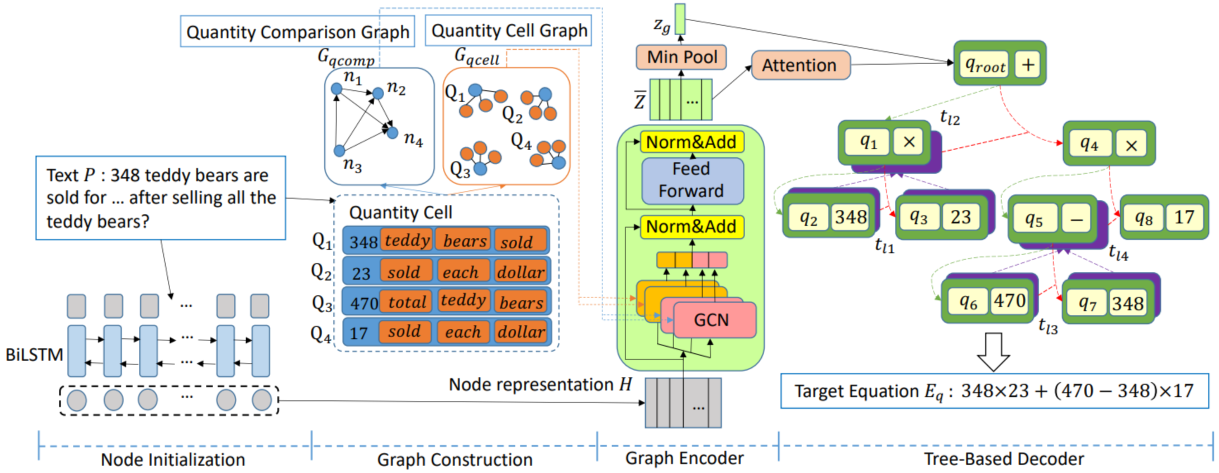
- **A seq2seq model with a well-designed tree-structured decoder (Xie and Sun, 2019)**
  - To use the structural information from expressions more effectively
  - Generate the pre-order sequence of a binary tree in a top-down manner



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

Background

- **Graph2Tree Solver (Zhang et al, 2020)**
  - Quantity Cell Graph
    - Capture relationships between quantities and their attributes for deep learning models
  - Quantity Comparison Graph
    - Loss of quantities' numerical qualities could be problematic for solution expressions
    - Retain the quantities' numerical qualities
  - Graph2Tree
    - Use a graph transformer to learn the latent quantity representations from our graphs
    - Use a tree structure decoder to generate a solution expression tree



Jipeng Zhang et al. Graphto-tree learning for solving math word problems. ACL. 2020.

Background

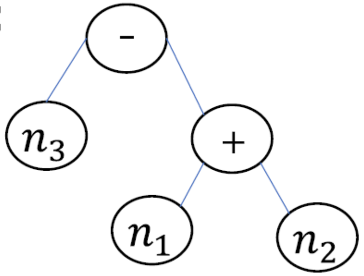
- **When a problem has multiple correct answer**

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

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

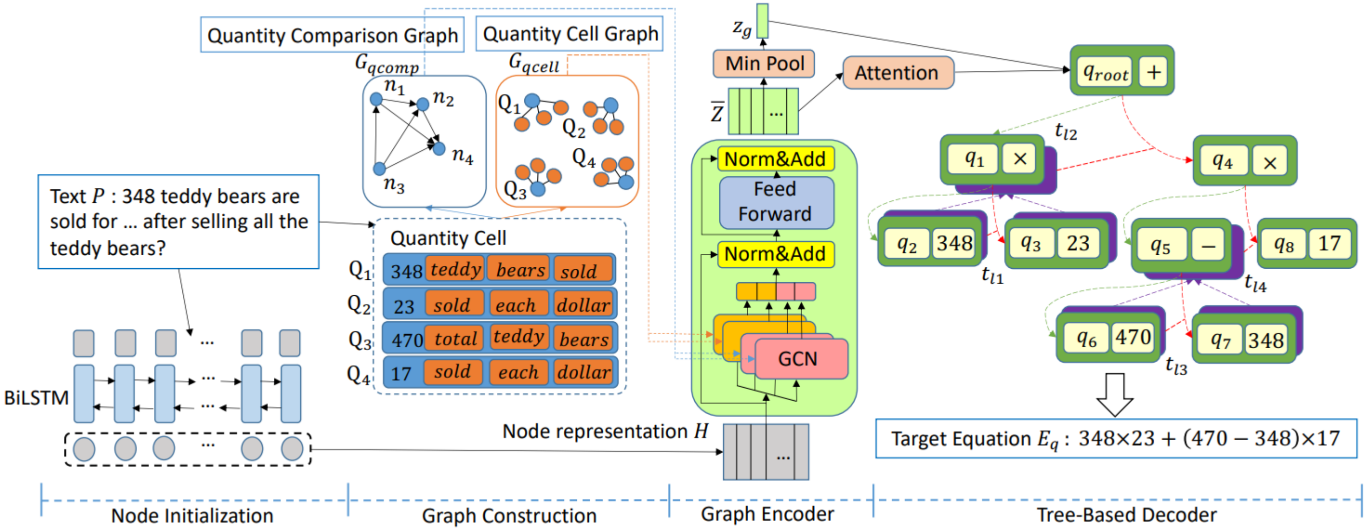
Table 3: Accuracy of template prediction module.

	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

- **Graph2Tree Solver (Zhang et al, 2020)**
  - Use multiple decoders to learn different expression sequences simultaneously
    - The tree decoder generates an equation following the pre-order traversal ordering
    - Subsequently, we generate the right child nodes recursively
  - The large & varying number of sequences for MWPS makes the strategy less adaptable



#Op	Pro (%)	AST-Dec (%)	GTS (%)	Our (%)
1	17.3	82.7	84.9	**85.5**
2	52.2	74.5	80.6	**83.7**
3	19.1	59.9	70.7	**71.7**
4	6.6	42.4	50.0	**51.5**
5	3.4	**44.1**	38.2	38.2
6	0.9	**55.6**	44.4	**55.6**

Table 5: Accuracy for increasing length of templates.

Jipeng Zhang et al. Graphto-tree learning for solving math word problems. ACL. 2020.

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)

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

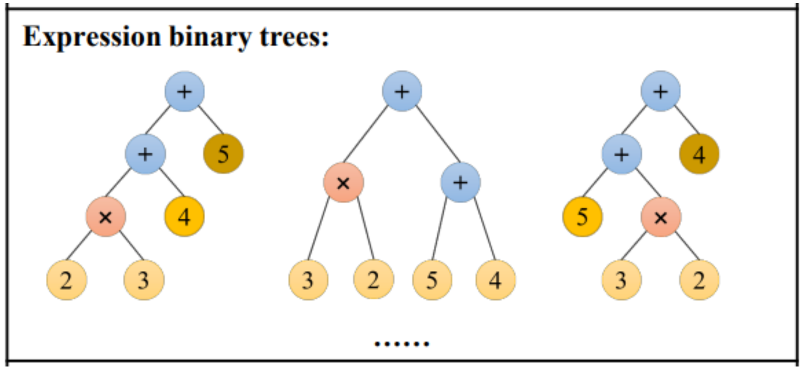
  - The uncertainty caused by the exchange of operands or sub-expressions
    - Addition & multiplication have the property that the operands or sub-expressions of both sides are allowed to be swapped

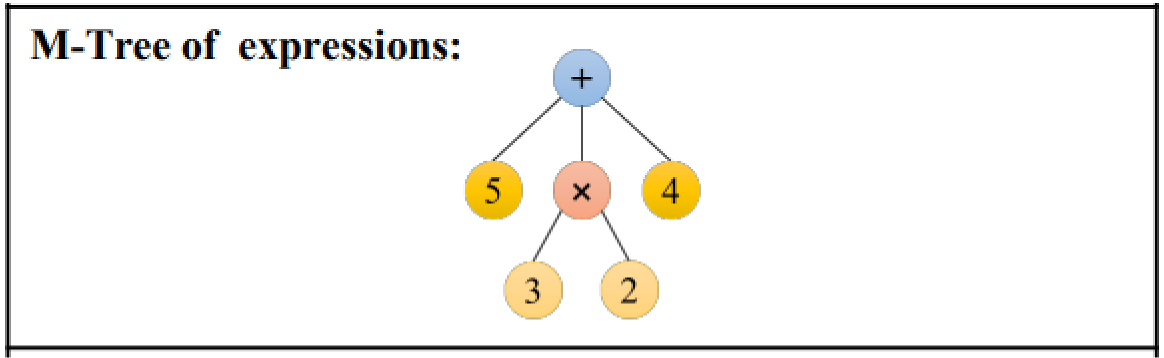
Introduction

- **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

Introduction

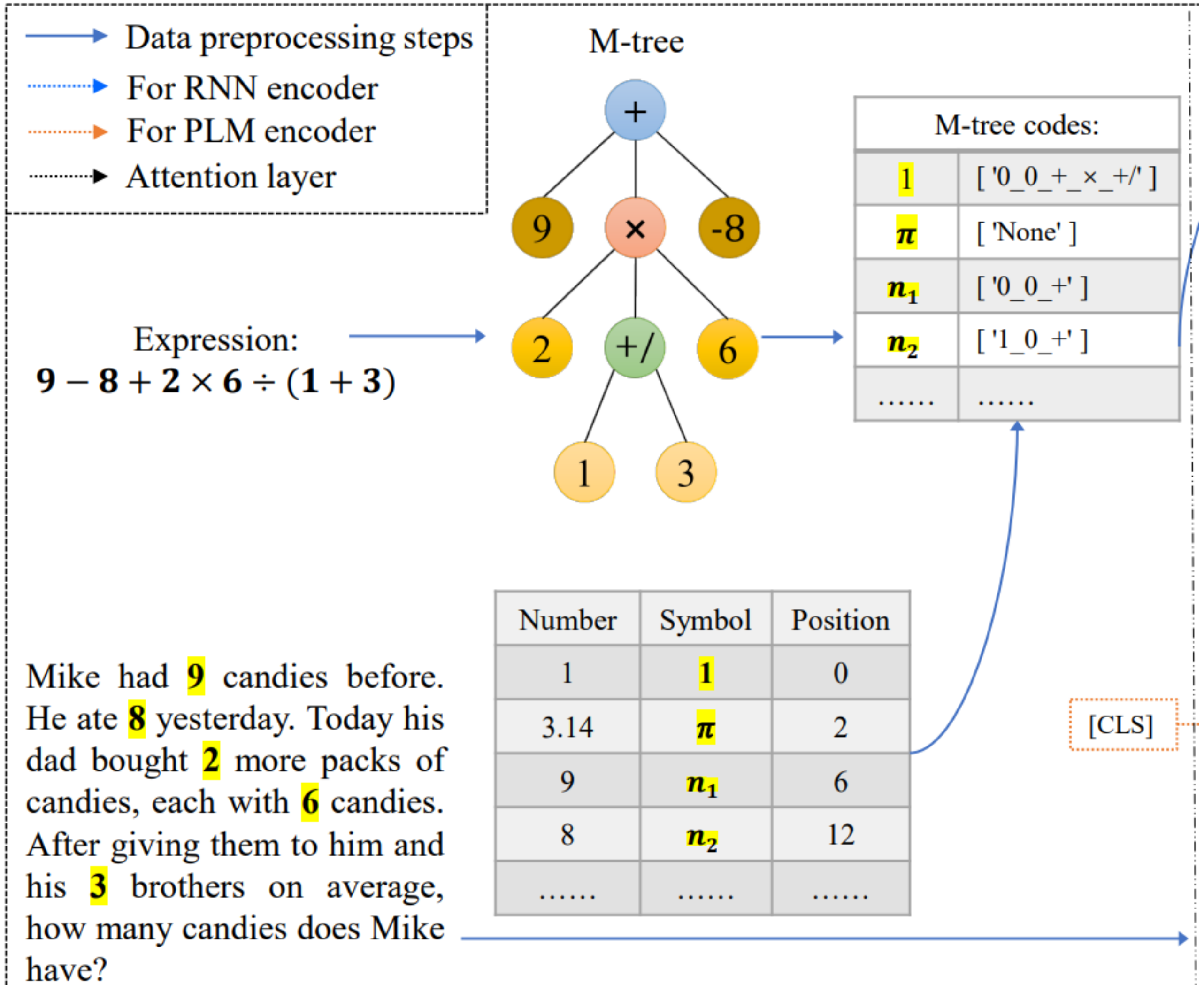
- **The M-tree unifies output structures in SUMC-Solver**
  - Deal with the uncertainty of computation orders for mathematical operations
    - Set the root to a specific operation and allow any number of branches for internal nodes in the M-tree
    - Reduce the diversity of the tree structure in the vertical direction
  - Deal with 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**

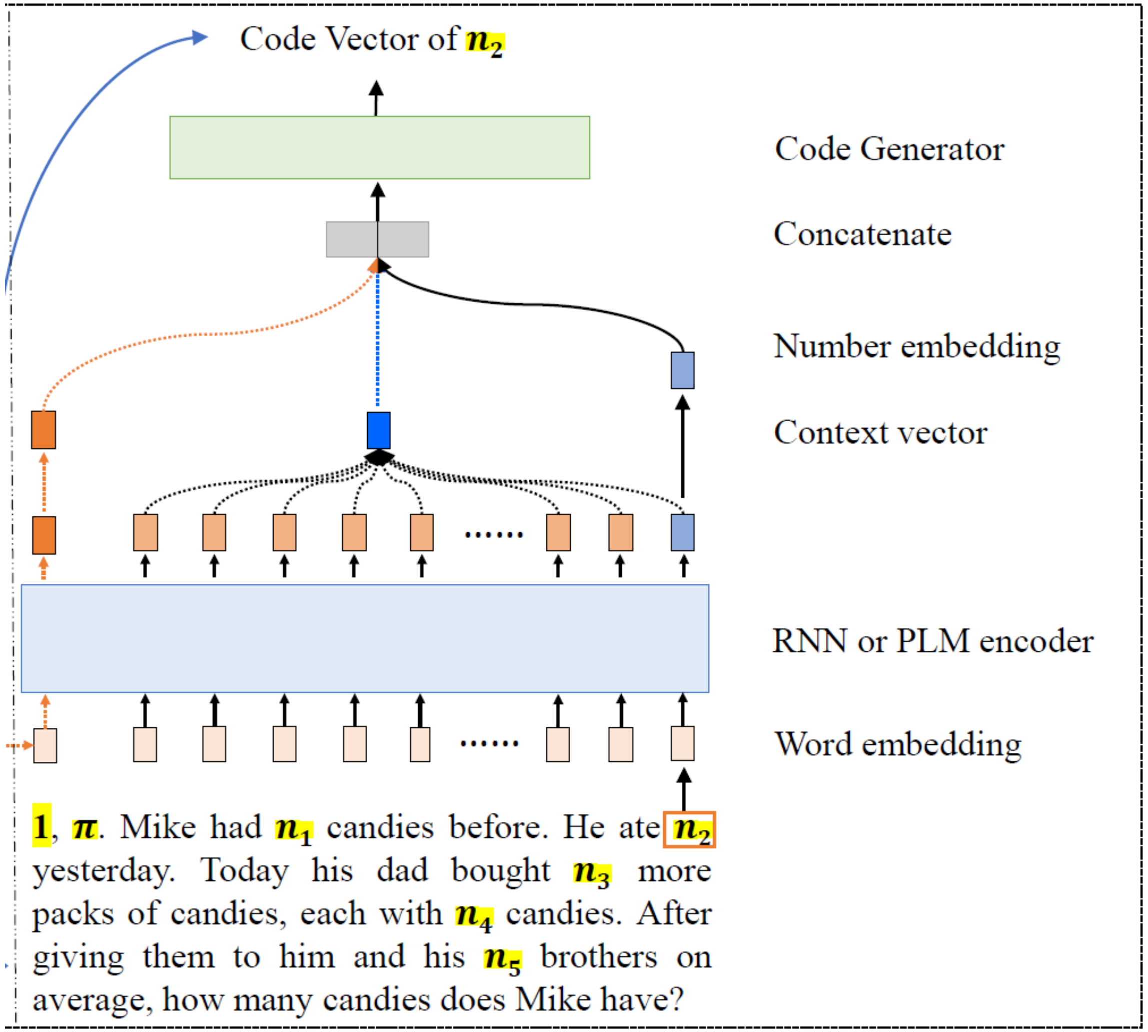


M-tree codes

- 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

Introduction

- **Contribution of the SUMC-Solver**
  - Analyze the causes of output diversity in MWP
  - Design a novel M-tree-based solution to unify the output

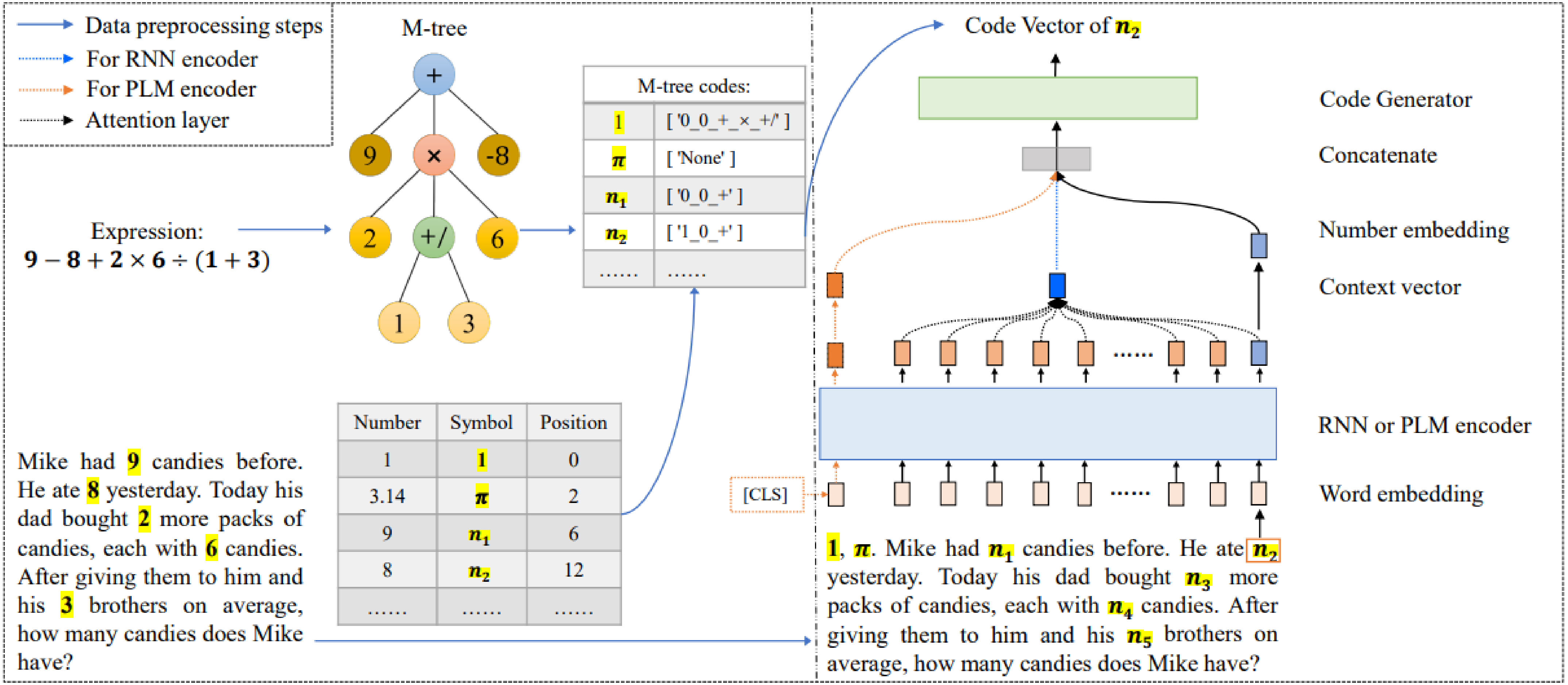
  - The first work to analyze mathematical expressions with M-tree codes and seq2code
    - Design the M-tree codes to represent the M-tree
    - Propose a seq2code model to generate the codes in a non-autoregressive fashion

  - Expriemental Result
    - Outperforms previous methods with similar settings in MAWPS & Math23K datasets
    - The case in low resource scenarios

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 $X = (x_1, x_2, ..., x_n)$
    - constants, including 1 and π , are required
  - 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

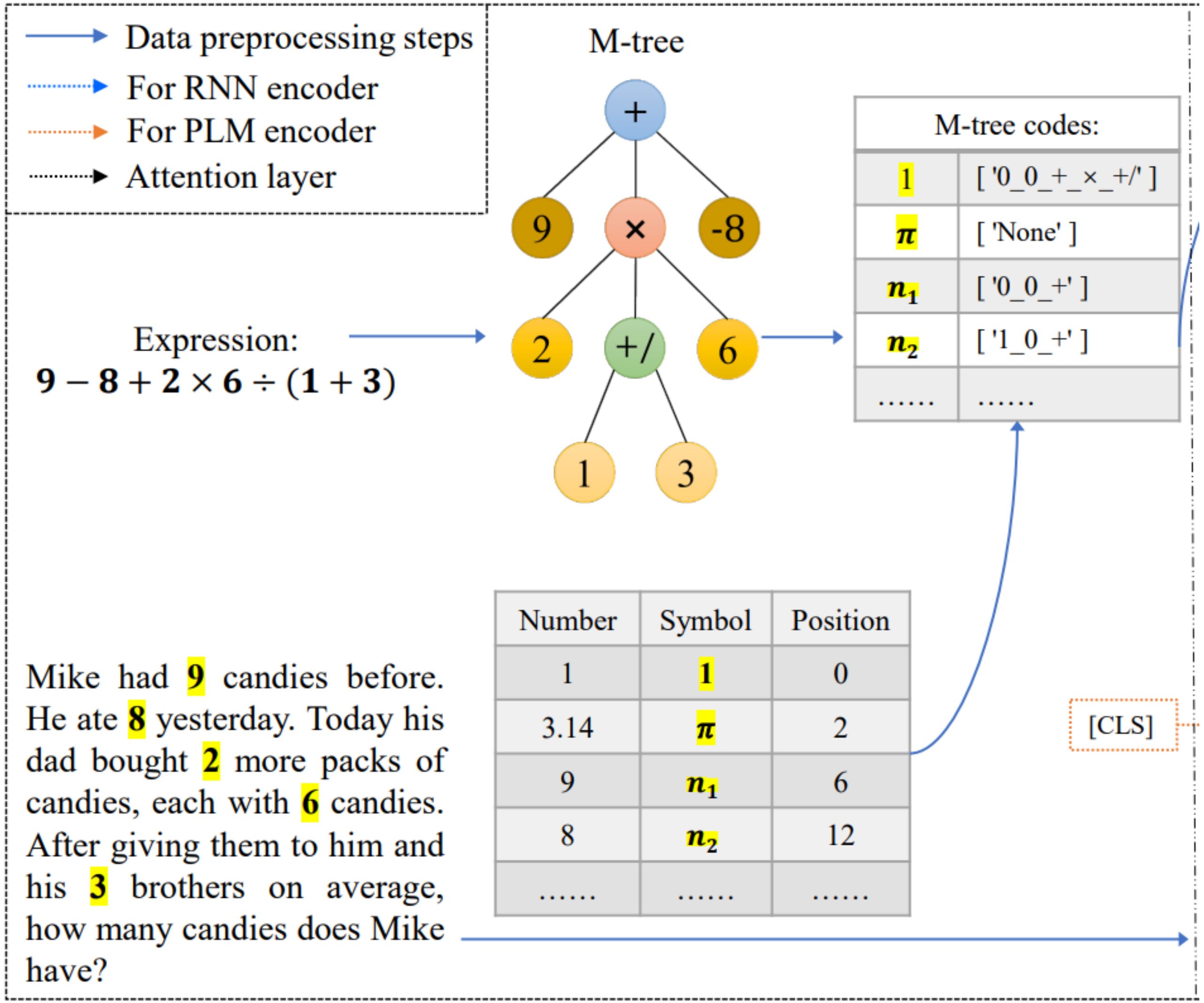
$$n_1 \times (n_2 \pm n_3) \longrightarrow n_1 \times n_2 \pm n_1 \times n_3$$

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

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

Method

- **M-Tree: The Design**



M-tree codes

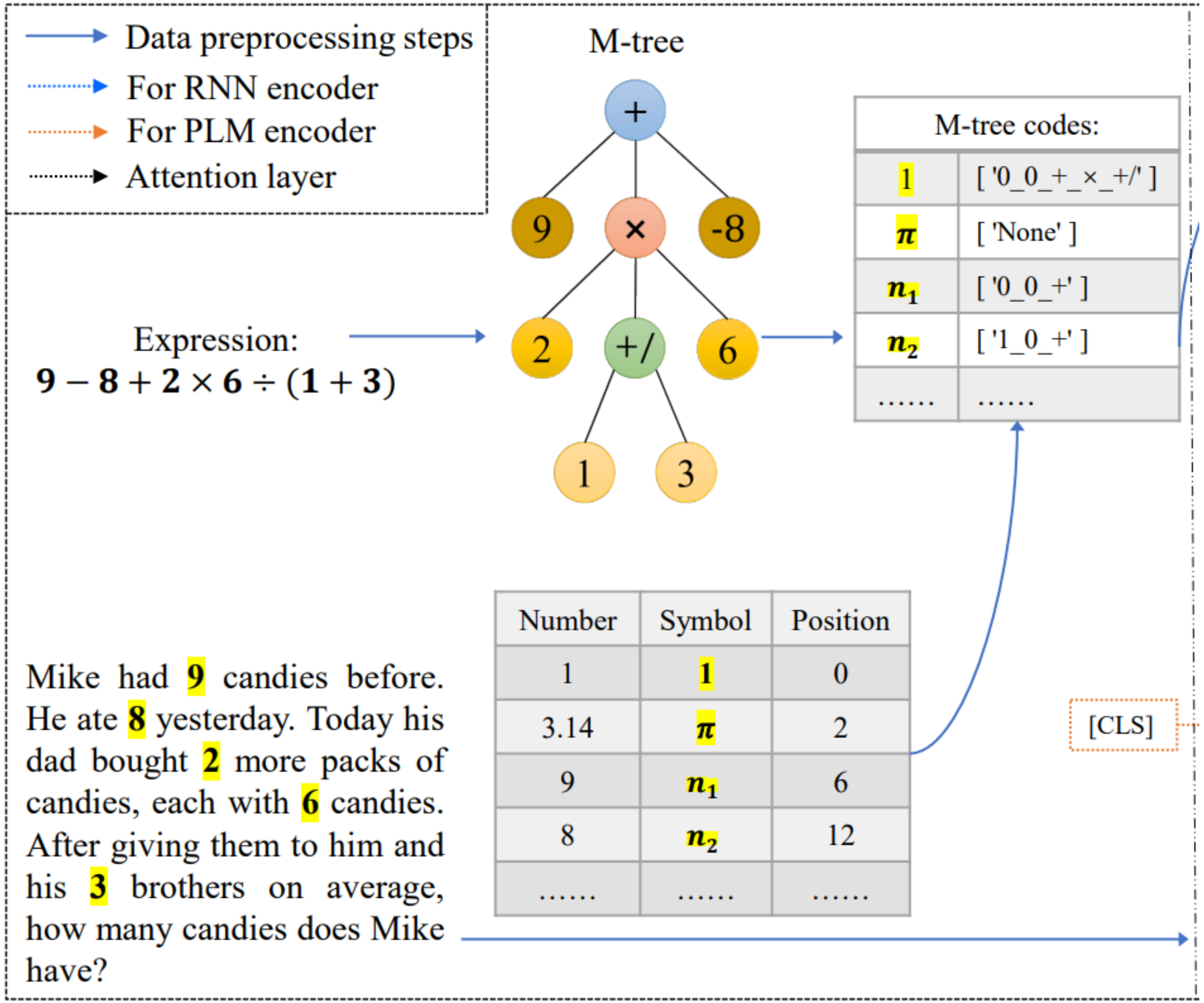
- Internal node
  - Each internal node has any $\mathbf{M}$ branches, where $\mathbf{M}$ is an integer greater than or equal to 1
  - Four types of internal nodes, corresponding to redefined operations $\{+, \times, \times-, +/\}$
- Leaf node
  - Four types of leaf nodes, corresponding to the numerical value: $\{v, -v, \frac{1}{v}, -\frac{1}{v}\}$

the original value v, the opposite of v
the reciprocal of v
the opposite of the reciprocal of 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 $\{v_1, v_2, \ldots, v_k\}$
- The node of "+" ("×") :
  $v_1 + v_2 +, \ldots, +v_k$ $(v_1 \times v_2 \times, \ldots, \times v_k)$
- The node of "×−"("+/") :
  $-v_1 \times v_2 \times, \ldots, \times v_k$ $\left(\frac{1}{v_1 + v_2 +, \ldots, +v_k}\right)$

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

$v_1 \div v_2 \rightarrow v_1 \times v_2'$		$v_1 - v_2 \times v_3 \rightarrow v_1 + v_2(\times-)v_3$	
$v_1 - v_2 \rightarrow v_1 + v_2'$		$v_1 \div (v_2 + v_3) \rightarrow v_1 \times v_2(+/)v_3$	

  - 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

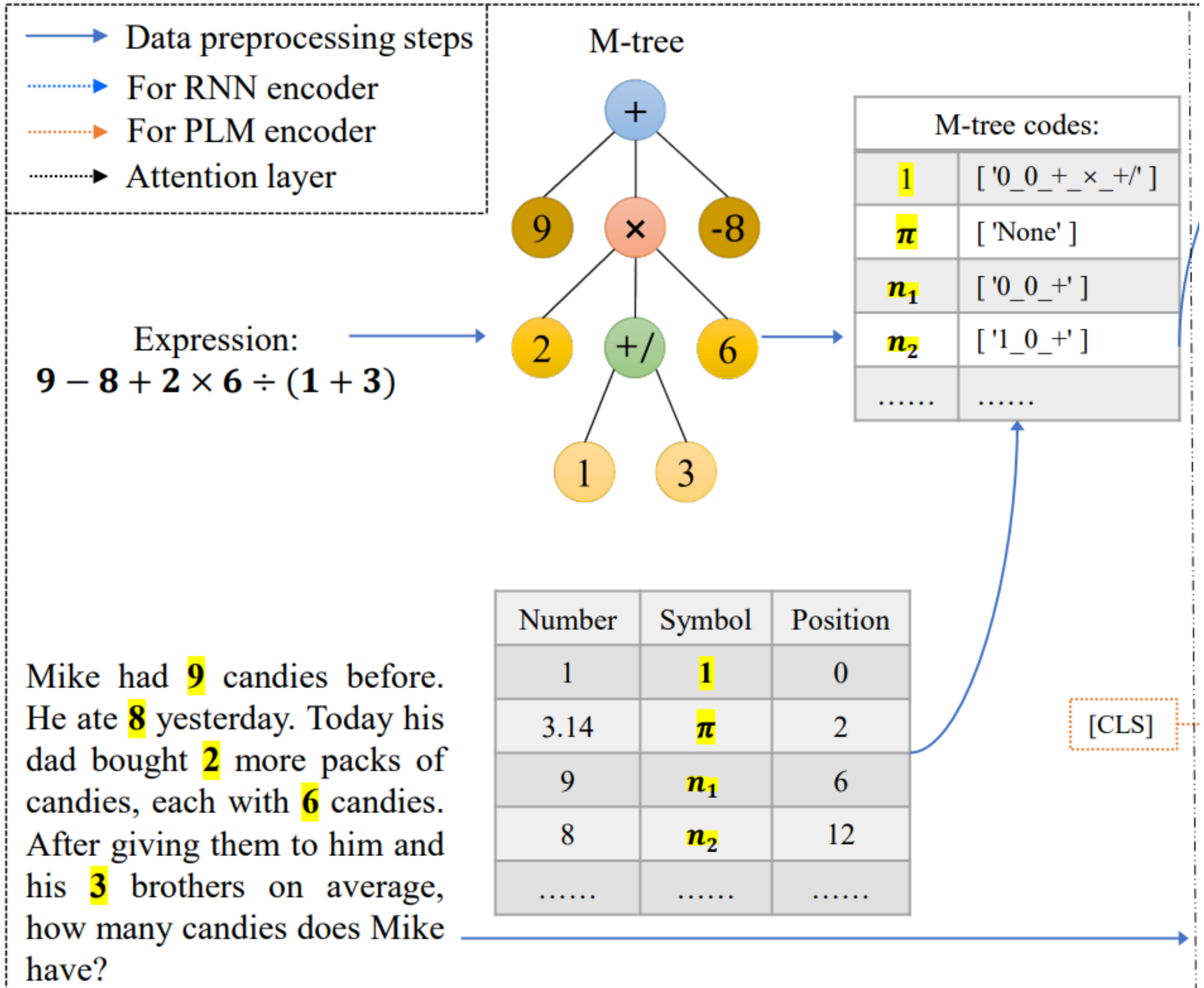
“$v_p = \times$ and $v_c = \times-$” $\rightarrow$ “$v_p = \times-$”

“$v_p = \times-$ and $v_c = \times-$” $\rightarrow$ “$v_p = \times$”

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

• Components of M-tree Codes



Each leaf node

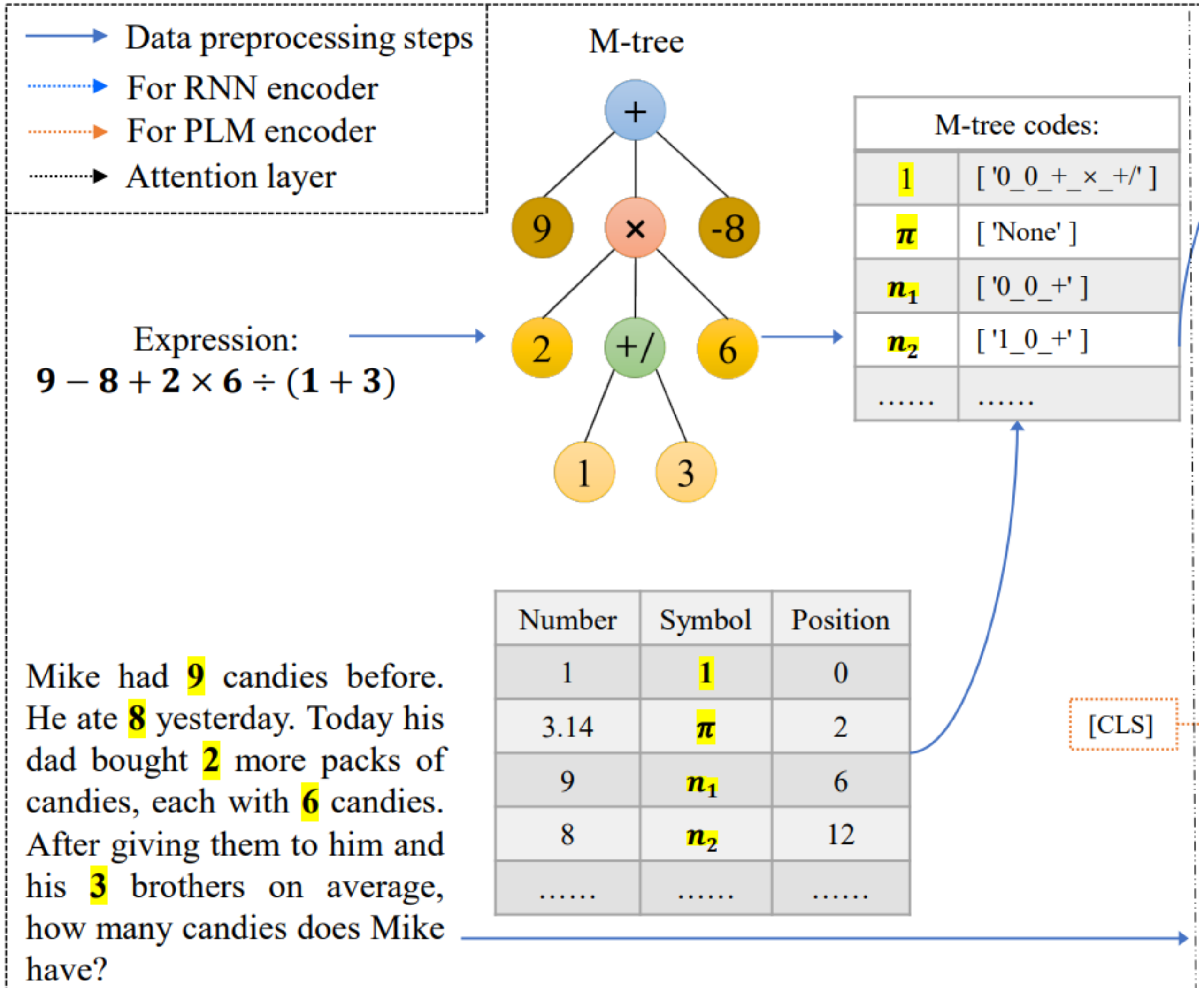
- The numerical value
  - If $v_i^{'} = v_i$, the code is set as "0_0";
  - If $v_i^{'} = -v_i$, the code is set as "1_0";
  - If $v_i^{'} = \frac{1}{v_i}$, the code is set as "0_1";
  - If $v_i^{'} = -\frac{1}{v_i}$, the code is set as "1_1";
- The sequential operation symbols of all internal nodes 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 l-dimensional non-one-hot vectors:
  - $\mathbf{C} = \{\mathbf{c_1}, \mathbf{c_2}, ..., \mathbf{c_m}\}$, the value of ci in the kth 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

- **Reducing M-tree codes to M-tree**
  - The process of converting M-tree to M-tree codes is reversible
  - A code vector is generated for each number in the text and mapped to one or more M-tree codes at first
  - The number is formatted according to the first part of the M-tree code
  - All the numbers are merged by scanning the second part of the M-tree code from back to front, while the M-tree is generated bottom-up

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 $\mathbf{H} = \{\mathbf{h}_1^x, \mathbf{h}_2^x, ..., \mathbf{h}_n^x\} \in \mathrm{R}^{n \times 2d}$

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

  - $\mathbf{e}_t^x$ is the word embedding vector
  - For the numerical value v_i in the problem
  - Its semantic representation $\mathbf{e}_i^c$ is modeled by the corresponding BiLSTM output vector $\mathbf{e}_i^c = \mathbf{h}_{q_i}^x$

Method

- **Sequence-to-Code Model: RNN encoder (Problem Encoder)**
  - Better capture the relationship between different numerical values and the relationship between vi and the unknown value to be solved (answer of the problem)

  - Use an attention layer to derive a context vector $\mathbf{E}_i$ for v_i
  - Context vector is expected to summarize the key information of the input problem
  - The context vector $\mathbf{E}_i$ is calculated as a weighted representation of the source tokens:

$$\mathbf{E}_i = \sum_t \alpha_{it} \mathbf{h}_t^x \quad \alpha_{it} = \frac{\exp\left(\operatorname{score}\left(\mathbf{e}_i^c, \mathbf{h}_t^x\right)\right)}{\sum_t \exp\left(\operatorname{score}\left(\mathbf{e}_i^c, \mathbf{h}_t^x\right)\right)}$$

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

  - $\mathbf{U}$ and $\mathbf{W}$ are trainable parameters

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

Method

- **Sequence-to-Code Model: PLM encoder (Problem Encoder)**
  - Encode the input sequence X to get the token embeddings $Ems = \{em_t^x\}_{t=1}^n$
  - Get the semantic representation $\mathbf{e}_i^c$ in the same way as the RNN encoder
  - Use the output embedding of the special token $[\mathrm{CLS}]$ in RoBERTa

$$\mathbf{e}_i^c = \mathbf{em}_{q_i}^x$$

$$\mathbf{E}_i = \mathbf{em}_{cls}^x$$

Method

- **Sequence-to-Code Model: Code Generator**
  - Use a simple three-layer FFNN to implement the generator
  - With the input $\mathbf{z}_i^c$, the final code vector $\mathbf{c}_i^{'}$ is generated
  - σ is an activation function

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

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

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

Method

- **Sequence-to-Code Model: Training Objective**
  - Given the training dataset $\mathbf{D} = \{(X^i, C^i) : 1 \leq i \leq N\}$

, where C^i is the set of all the code vectors corresponding to the numerical values appearing in X^i

where l is the dimensionality of code vector

$$\mathcal{L} = \sum_{(X^i, C^i) \in \mathbf{D}} \sum_{\mathbf{c}_i \in C^i} \mathcal{L}_{MSE}(\mathbf{c}_i, \mathbf{c}_i^{'})$$

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

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

- **Compared Method**
  - T-RNN (Wang et al. (2019))
    - A seq2seq model to predict a tree-structure template, includes inferred numbers and unknown operator, a RNN to obtain unknown operator nodes in a bottom-up manner
  - StackDecoder (Chiang and Chen (2019))
    - The RNN to understand the semantics of problem, a stack was applied to generate post expression
  - GTS (Xie and Sun (2019))
    - A RNN to encode the input and another RNN to generate the expression based on top-down decomposition and bottom-up subtree embedding
  - GTS-PLM
    - Replace the encoder with a pre-trained language model compared to the original GT

Experiment

- **Compared Method**
  - SAU-Solver (Qin et al. (2020))
    - Universal Expression Trees to handle MWPs with multiple unknowns and equation
    - Encode the input and a well-designed decoder considering the semantic transformation between equations obtains the expression
  - Graph2Tree (Zhang et al., 2020b)
    - A graph-to-tree model that leverages an external graph-based encoder to enrich the quantity representations in the problem
  - UniLM-Solver UNIfied Pre-trained Language Model (UniLM) (Dong et al., 2019)
    - Use to model the generation process from the input text to the output expressio

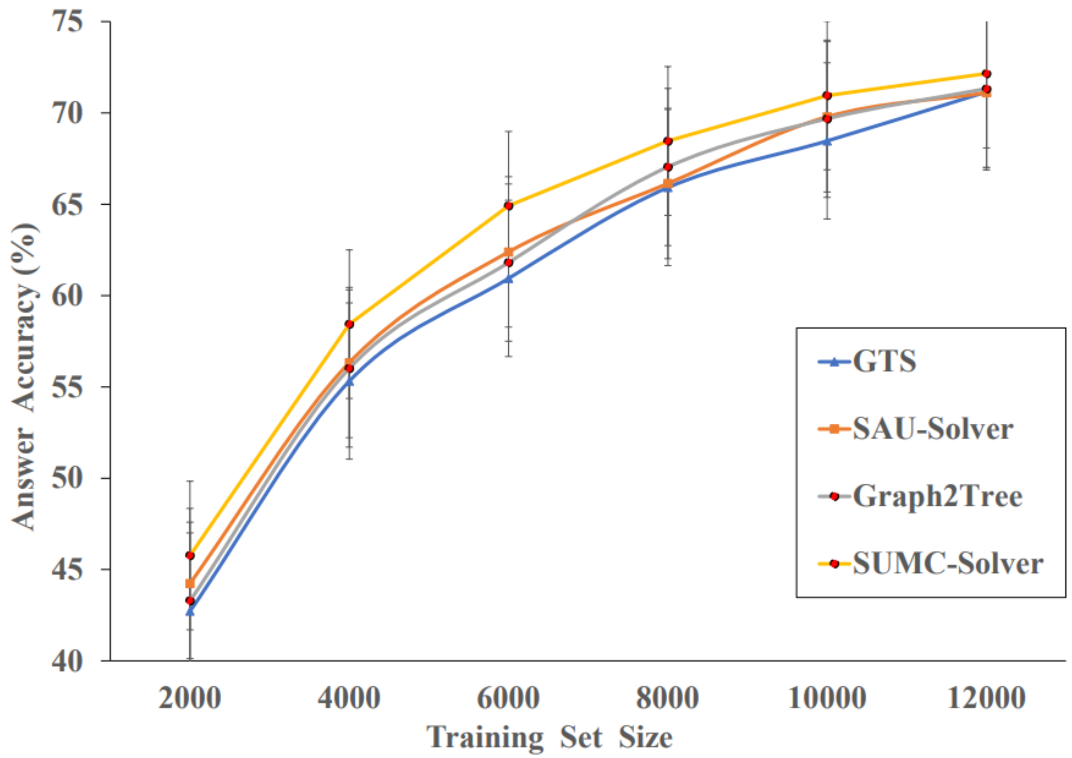
Experiment

- **Answer Accuracy**
  - SUMC-Solver outperforms all baselines in the two MWP datasets
    - M-tree: SUMC-Solver
    - An RNN as the encoder: StackDecoder & T-RNN
    - The binary-tree: GTS, SAU-Solver, Graph2Tree
    - A PLM as the encoder: UniLM-Solver

	Model	Math23K	MAWPS*
RNN	T-RNN	66.9	66.8
	StackDecoder	67.8	-
	GTS	75.6	$75.2^{\dagger}$
	SAU-Solver	$76.2^{\dagger}$	$75.5^{\dagger}$
	Graph2Tree	$76.6^{\dagger}$	$78.1^{\dagger}$
	SUMC-Solver	**77.4**	**79.9**
PLM	UniLM-Solver	$77.5^{\dagger}$	$78.0^{\dagger}$
	GTS-PLM	$79.5^{\dagger}$	$79.8^{\dagger}$
	SUMC-Solver	**82.5**	**82.0**

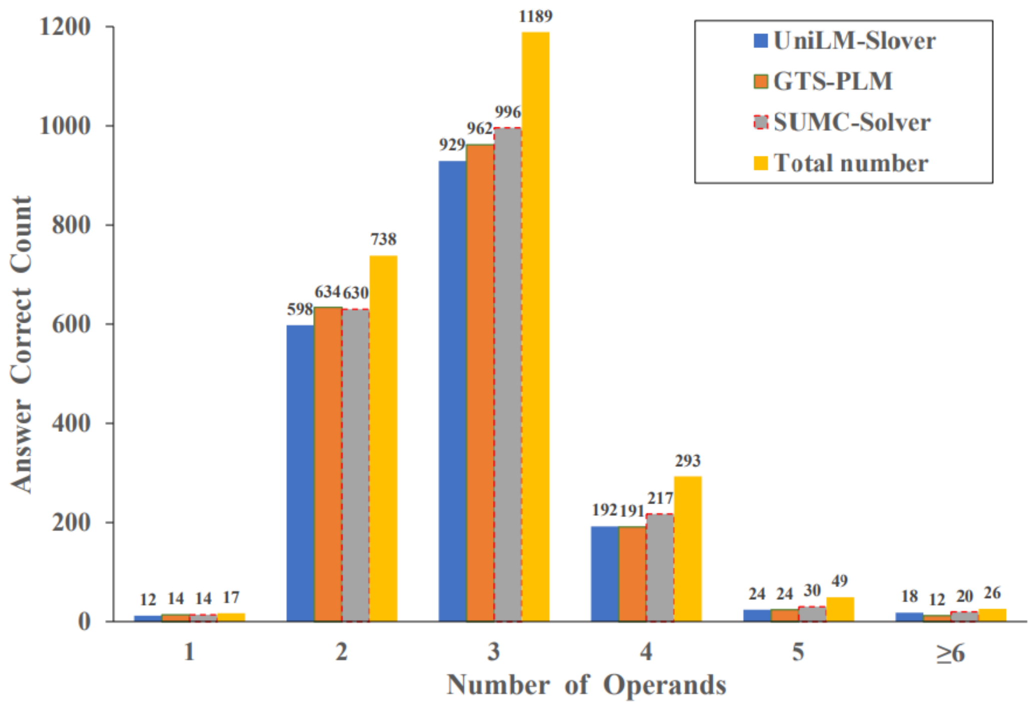
Experiment

- **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



Conclusion

- **Structure-Unified M-Tree Coding Solver (SUMC-Solver)**
  - Apply the M-tree to unify the diverse output and the seq2code model to learn the M-tree
  - Experimental results
    - Outperform several state-of-the-art models under similar setting
    - Perform much better under low-resource conditions

Conclusion

- **Limitations**
  - Some special Mtrees need to be distinguished by introducing special symbols randomly when converting them into M-tree codes, which makes the M-tree codes correspond to the MWP may not be unique
    - About 90% of the data do not belong to this particular case
    - About 10%, despite the increased difficulty, they are still learnable based on previous work experience, which makes SUMC-Solver still achieve a significant performance improvement
  - The network structure is relatively simple for the seq2code framework used in SUMC-Solver

    Previous Work

    - The use of graph-based encoders and the introduction of external knowledge to enrich the representation of the input problem
    - Seq2code can be naturally integrated with these improved approaches to try to achieve better results