A Goal-Driven Tree-Structured Neural Model for Math Word Problems

IJCAI 2019

Zhipeng Xie*†, Shichao Sun†

Shanghai Key Laboratory of Data Science, Fudan University

School of Computer Science, Fudan University

Part 1.

Background

- **Math word problems (MWPs)**
  - Automatically answer a mathematical query according to the text description
  - Typical MWP
    - Short narrative that describes a partial state of the world and poses a question about an unknown quantity

A typical math word problem

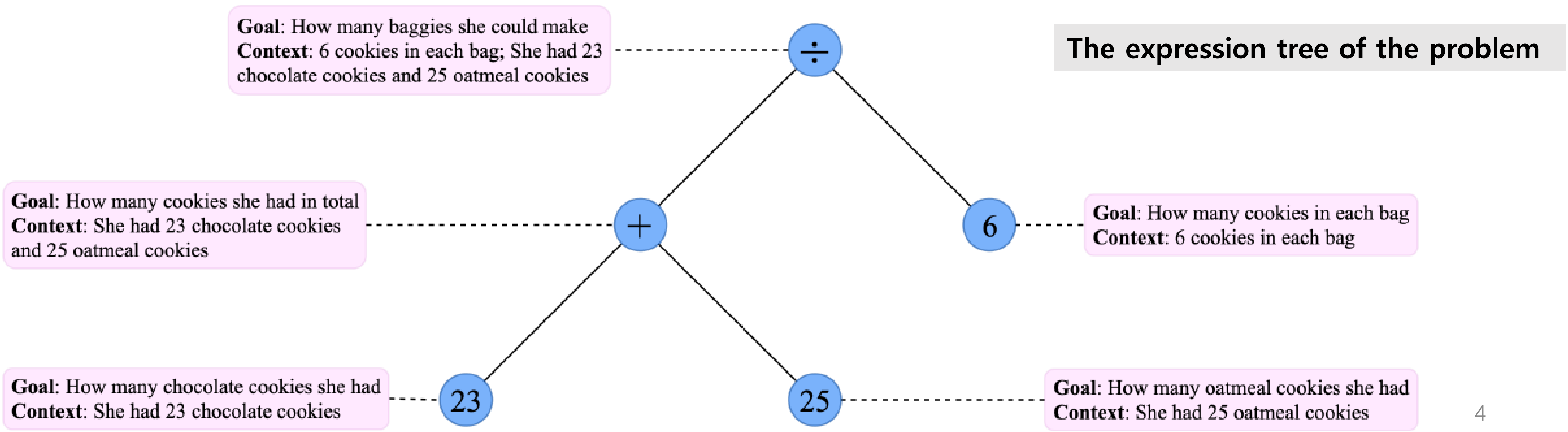
Problem: Robin was making baggies of cookies with 6 cookies in each bag. If she had 23 chocolate cookies and 25 oatmeal cookies, how many baggies could she make?
Solution Expression: $(23 + 25) \div 6$ **Solution:** 8

Background

- **Seq2Seq-based solver**
  - Pros
    - The power of generating new expressions that do not exist in the training dataset
    - The Seq2Seq-based models exists in that they do not rely on hand-crafted features
  - Cons
    - Do not match the goal-driven mechanism in human problem solving
    - To model the tree-structured relationship of expression tree through its post order traversal sequence during decoding

Background

- **Seq2Seq-based solver**
  - Human Problem Solving
    - Decompose the goal into two sub-goals combined by an operator based on the relevant information
    - Decompose the goal recursively for solving a math word problem and finally generates an expression tree



Background

- **Goal-driven mechanism in human problem solving**
  - Design a novel model to generate expression tree

  - Process
    - Model firstly initializes the root goal vector which represents the final goal of the problem
    - Summarizes relevant information of the problem into the context vector
    - A token is predicted using the goal vector and its context vector, which implicitly decides whether the goal should be decomposed further
    - Prediction and the goal decomposition process are repeated for them

  - For a commutative operator such as “+” or “×”
    - Its right sub-goal may be the same as the left one
    - Due to its commutative property

  - To address this issue
    - Our model completes the construction of the left subtree before generating the right sub-goal
    - The generation of right sub-goal takes the information of its left sibling subtree into consideration, which is encoded as a subtree embedding by a recursive neural network

Introduction

- **A Goal-Driven Tree-Structured Neural Model for Math Word Problems**
  - Neural model to generate an expression tree in a human-like goal-driven way for solving math word problems
    - The first tree-structured neural model for MWPs
  - The information explicitly flows through the expression tree
    - Top-down (goal decomposition) manners
    - Bottom-up (subtree embedding) manners
  - Experimental results
    - Significantly outperforms several state-of-the-art systems on the dataset Math23K

Introduction

- **Related Work**
  - Rule-based methods
  - Statistical machine learning methods
  - Semantic parsing methods

  - Deep learning methods
    - Seq2seq model with Recurrent Neural Network (RNN) in its encoder and decoder
    - Convolutional Neural Network (CNN) instead of RNN

  - Huang et al. [2018]
    - The Seq2Seq model may generate spurious numbers or predict numbers at wrong positions
    - Copy-and-alignment mechanism to the standard Seq2Seq model to solve these issues

  - Wang et al. [2018a]
    - Seq2Seq model always suffers from an equation duplication problem: a MWP can be solved by multiple expressions
    - An equation normalization method to solve this problem

Problem Statement

- **Problem text** P
  - A sequence of word tokens and numeric values
  - Usually begins by describing a partial quantitative state of a world, followed by simple updates
  - Ends with a query about an unknown quantity

- **The ordered list of numeric values in** P **according to their order in the problem text** n_P
  - At a preprocessing step
    - All the number tokens are treated as a special word token NUM
    - Usually do not care about their exact values in solving a math word problem

Problem Statement

- **Solution expression tree T**
  - Mathematical expression tree
    - Can be easily transformed from the solution expression
  - Capture the relations among these numeric values which are described or implied literally by the problem text
  - T may contain constant quantities, mathematical operators, and numeric values in n_P from problem text P

Problem Statement

- **The set of mathematical operators** V_{op}

- **The set of constant quantities** V_{con} π 2
  - Special numeric values that may occur in the solution but not in the problem text

- **The target vocabulary of** P $V^{dec} = V_{op} \cup V_{con} \cup n_P$
  - Two identical numbers since the number occurs in two different positions of P
  - Choose the occurrence of higher probability (Equation (8)) as the target

$$\text{prob}(y|\mathbf{q}, \mathbf{c}, P) = \frac{\exp\left(\text{s}\left(y|\mathbf{q}, \mathbf{c}, P\right)\right)}{\sum_i \exp\left(\text{s}\left(y_i|\mathbf{q}, \mathbf{c}, P\right)\right)}$$

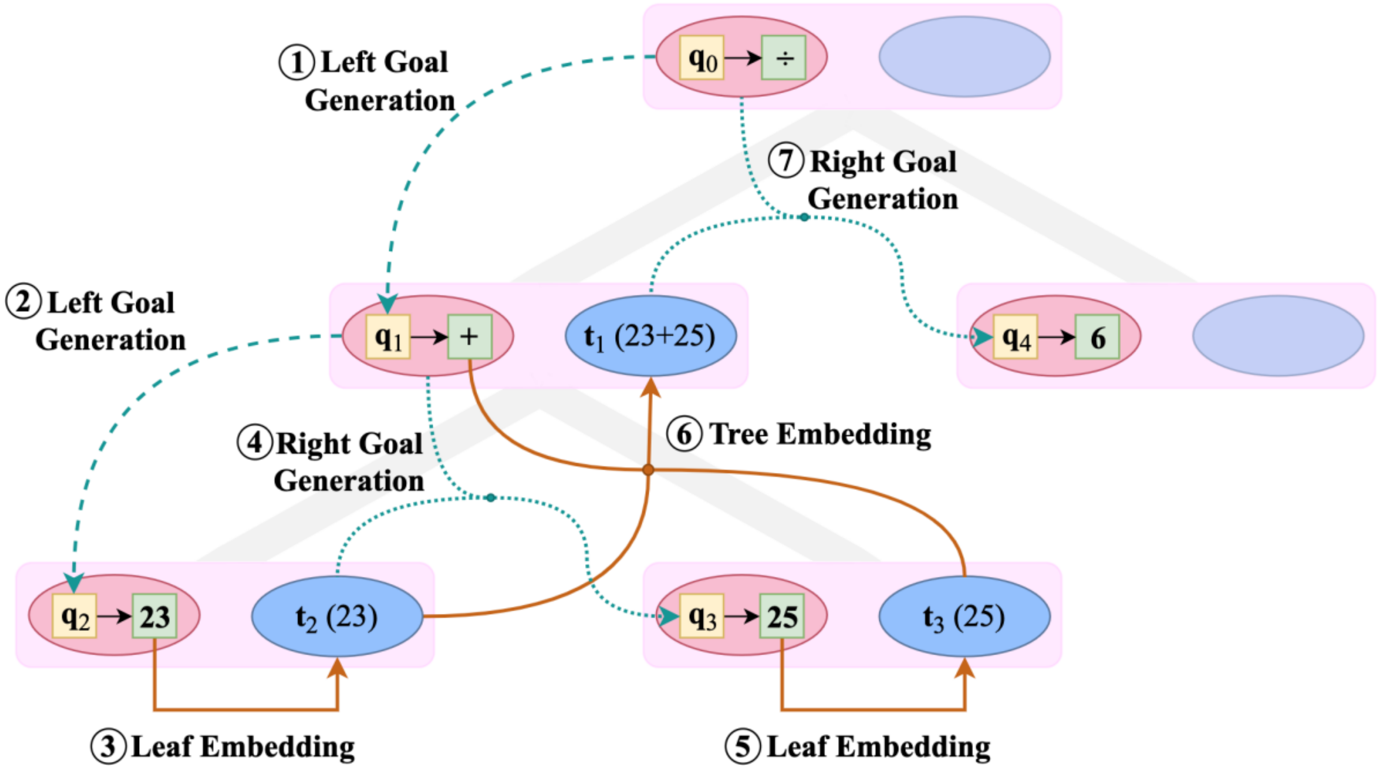
Model Description

- **Each node $\mathbf{n}$ in the expression tree T**
  - The goal vector $\mathbf{q}$
    - Instruct how the subtree root from node $\mathbf{n}$ should be constructed
    - The subtree is generated to realize the goal
  - First predicts the token $\hat{y}$ according to the goal vector $\mathbf{q}$
  - Predicted token naturally decides whether the goal should be decomposed further
    - If the predicted token is a mathematical operator, the goal will be decomposed into two sub-goals (a left sub-goal $\mathbf{q}_l$ and a right one $\mathbf{q}_r$)
    - The left (right) sub-goal serves to drive the construction of the left (right) subtree of $\mathbf{n}$
  - The goal will be simply realized by the predicted numeric value or constant quantity
    - Such a goal decomposition process is conducted recursively just like a depth-first traversal

Model Description

- **The goal decomposition process**
  - The left sub-goal is generated according to the goal vector and the predicted token of its parent node

Goal-driven Tree-structured Model



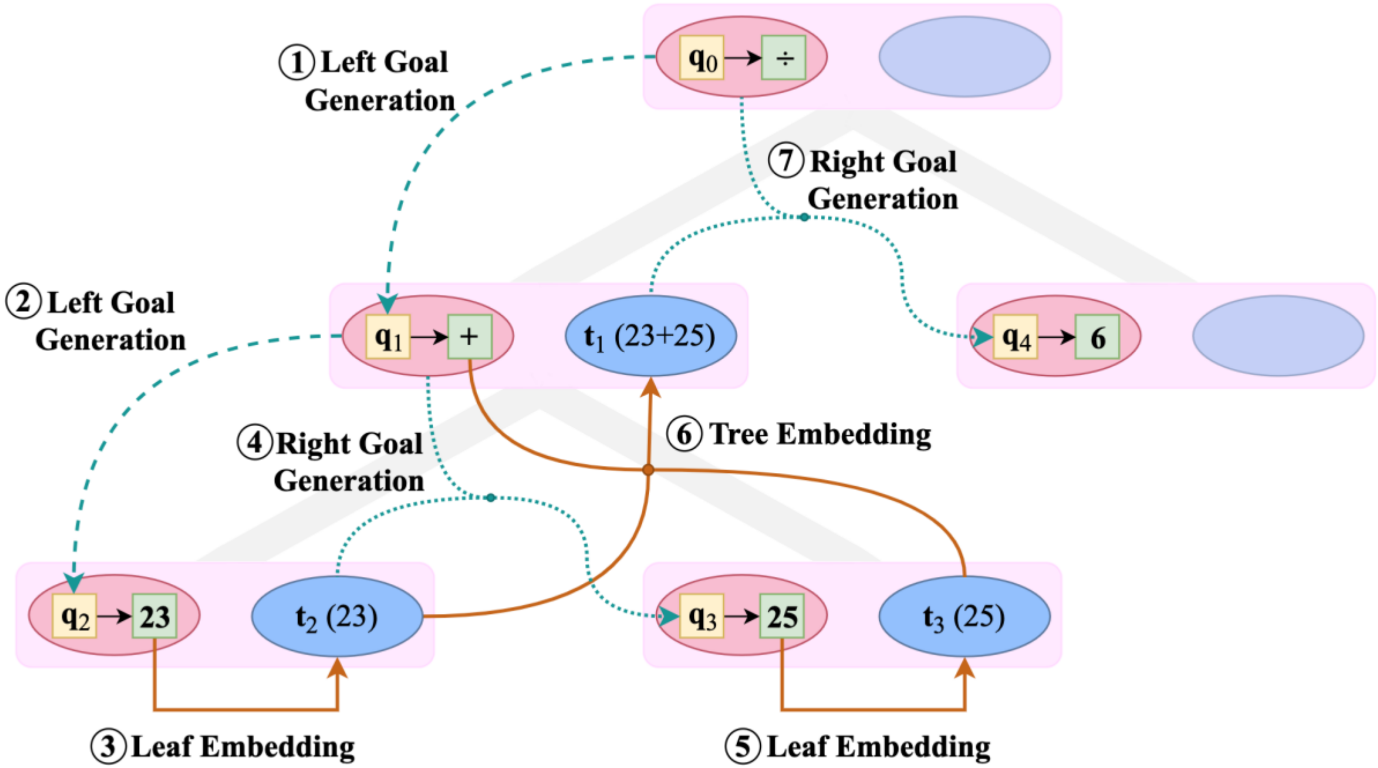
Left Sub-Goal Generation

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

Model Description

- **The goal decomposition process**
  - Take the information of its left sibling subtree into consideration, in addition to the parent goal and the left sub-goal

Goal-driven Tree-structured Model



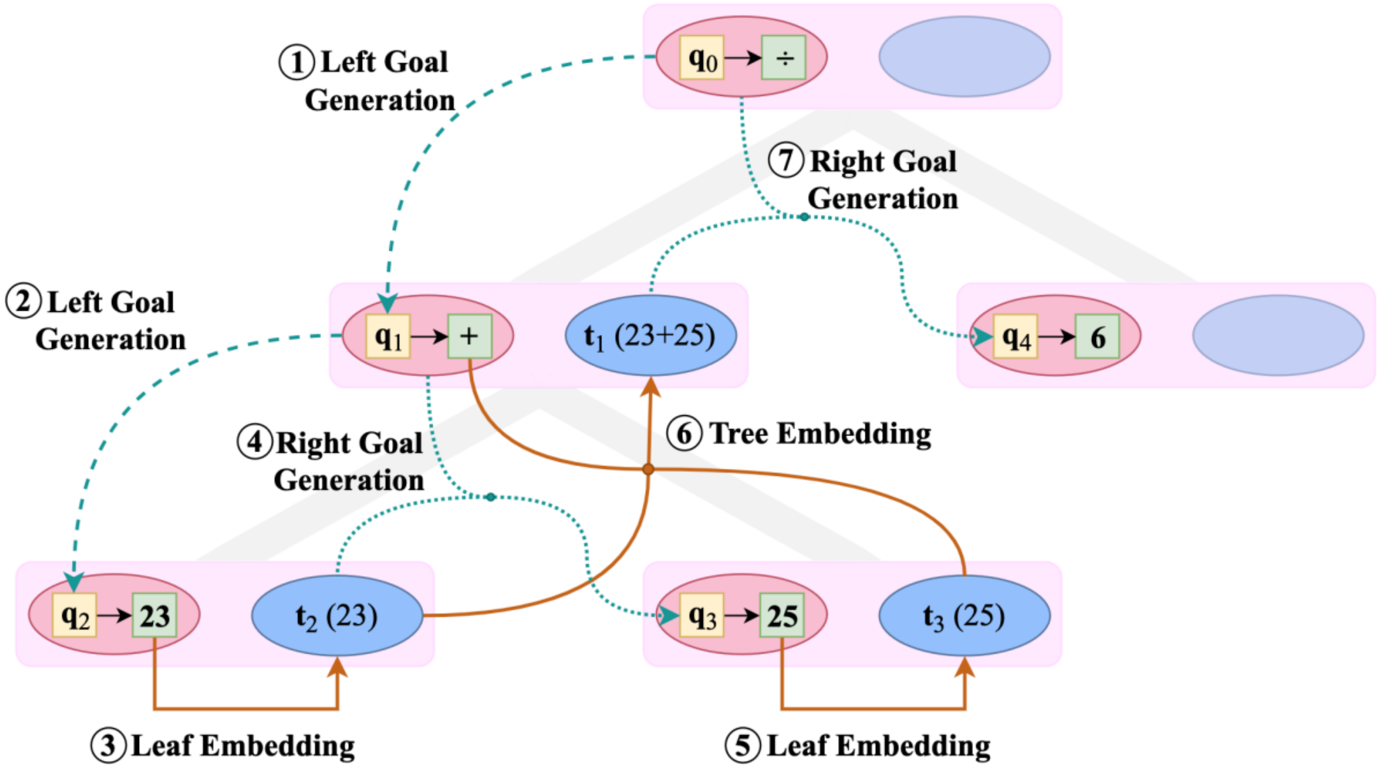
Right Sub-Goal Generation

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

Model Description

- **The goal decomposition process**
  - Encode the subtree information of a non-leaf node, bottom-up RNN is defined which fuses the token embedding of its mathematical token, and the embeddings of its left and right subtrees

Goal-driven Tree-structured Model



Subtree Emebedding via RNN

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

$$comb(\mathbf{t}_l, \mathbf{t}_r, \hat{y}) = g_t \odot C_t$$

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

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

Model Description

- **Encoder**
  - An input problem text $P = x_1 x_2 \ldots x_n$
  - Each word token x_i is firstly transformed into the corresponding word embedding $\mathbf{x}_i$
    - Look up an encoder embedding matrix $\mathbf{M}_{sen}$
  - The sequence of embeddings is inputted to the Gated Recurrent Unit (GRU) [Cho et al., 2014]
  - The function of a two-layer GRU
    - Produce a sequences of encoder hidden states one-by-one
    - The final hidden state $\mathbf{h}_s^p$ has incorporated contextual information of the source token

$$\overrightarrow{\mathbf{h}_s^p} = \mathrm{GRU}(\overrightarrow{\mathbf{h}_{s-1}^p}, \mathbf{x}_s) \quad \overleftarrow{\mathbf{h}_s^p} = \mathrm{GRU}(\overleftarrow{\mathbf{h}_{s+1}^p}, \mathbf{x}_s) \quad \mathbf{h}_s^p = \overrightarrow{\mathbf{h}_s^p} + \overleftarrow{\mathbf{h}_s^p}$$

Kyunghyun Cho et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. EMNLP. 2014.

Model Description

- **Root Goal Initialization and Token Embedding**
  - The top-down goal decomposition process
    - Initialize the goal vector $\mathbf{q}_0$ of the root node $\mathbf{n}_0$
    - According to the hidden states of the encoder of P
    - The final hidden states of forward/backward sequence $\overrightarrow{\mathbf{h}_n^p}$ $\overleftarrow{\mathbf{h}_0^p}$

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

  - Token embedding

$$\mathbf{e}(y|P) = \begin{cases} \mathbf{M}_{op}(y) & \text{if } y \in V_{op} \\ \mathbf{M}_{con}(y) & \text{if } y \in V_{con} \\ \mathbf{h}^p_{loc(y,P)} & \text{if } y \in n_P \end{cases}$$

Model Description

• Top-down Goal Decomposition

- Context vector $\mathbf{c}$
  - Given a goal vector $\mathbf{q}$, It summarizes relevant information of the problem at hand, which is expected to help predict the token and make the following decisions
  - Weighted representation of the source tokens by a vector $\mathbf{a}$ of attention weights a_s

$$\mathbf{c} = \sum_s a_s \mathbf{h}_s^p \qquad a_s = \frac{\exp(\text{score}(\mathbf{q}, \mathbf{h}_s^p))}{\sum_i \exp(\text{score}(\mathbf{q}, \mathbf{h}_i^p))}$$

$$\text{score}(\mathbf{q}, \mathbf{h}_s^p) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{q}, \mathbf{h}_s^p])$$

◦ Trainable parameters $\mathbf{v}_a$ $\mathbf{W}_a$

- Unnormalized log probability of generating a token y from the target vocabulary V^{dec}

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

◦ Trainable vector $\mathbf{w}_n$
◦ Trainable matrix $\mathbf{W}_s$
◦ Token embedding of y $\mathbf{e}(y|P)$

Model Description

- **Top-down Goal Decomposition**
  - The normalization of $\mathrm{s}(y|\mathbf{q},\mathbf{c},P)$ through softmax over target vocabulary

$$\mathrm{prob}(y|\mathbf{q},\mathbf{c},P) = \frac{\exp\left(\mathrm{s}\left(y|\mathbf{q},\mathbf{c},P\right)\right)}{\sum_i \exp\left(\mathrm{s}\left(y_i|\mathbf{q},\mathbf{c},P\right)\right)}$$

  - The predicted token implies a decision about how to realize the goal
    - If $\hat{y}$ is a numeric value or a constant quantity, the goal is realized directly by $\hat{y}$
    - Otherwise (i.e., $\hat{y}$ is an operator), the goal will be decomposed into two sub-goals

$$\hat{y} = \arg\max_{y \in V^{dec}} \mathrm{prob}(y|\mathbf{q},\mathbf{c},P)$$

Model Description

- **Left Sub-Goal Generation**
  - Predicted token $\hat{y}$ is an operator
    - The current goal $\mathbf{q}$ will be realized by a left(right) sub-goal $\mathbf{q}_l$ $\mathbf{q}_r$
  - Left sub-goal is calculated by a two-layer feedforward neural network with gating mechanism

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

- Trainable matrices $\mathbf{W}_{ol}$ $\mathbf{W}_{cl}$ $\mathbf{W}_{gl}$ $\mathbf{W}_{le}$
- The hidden state $\mathbf{h}_l$
  - Parent node delivers to its left child

Model Description

- **Right Sub-Goal Generation**
  - The right sub-goal quantity of right child takes into account the left child subtree
    - The left child subtree has been generated prior to the right child owing to the essence of pre-order traversal
    - The left subtree is encoded bottom up as $\mathbf{t}_l$ according to the recursive neural network

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

- Trainable matrices $\mathbf{W}_{or}$ $\mathbf{W}_{cr}$ $\mathbf{W}_{gr}$ $\mathbf{W}_{re}$
- The tree embedding of its sibling node $\mathbf{t}_l$

Model Description

• Subtree Emebedding via Recursive Neural Network

- A recursive neural network to encode a subtree in a bottom-up manner
- The embedding $\mathbf{t}$ of t is defined recursively as:

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

◦ Subtree at hand t

Case 1 : If the predicted token is an operator

- $(\hat{y} \in V_{op})$ means the subtree t must have two child subtrees t_l t_r
- The embedding of t needs to fuse the information from the operator $\hat{y}$, the child
- As done by the function $comb(\mathbf{t}_l, \mathbf{t}_r, \hat{y})$ with gating mechanism

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

◦ Trainable parameter matrices $\mathbf{W}_{ct}$ $\mathbf{W}_{gt}$

◦ The embedding of the operator $\mathbf{e}(\hat{y}|P)$

Model Description

- **Subtree Emebedding via Recursive Neural Network**

Case 2 : If $\hat{y}$ is a numeric value or a constant quantity

- The recursion stops and t becomes a leaf node
- The embedding of subtree t is simply set as the corresponding token embedding

$\mathbf{e}(\hat{y}|P)$

$$\mathbf{e}(y|P) = \begin{cases} \mathbf{M}_{op}(y) & \text{if } y \in V_{op} \\ \mathbf{M}_{con}(y) & \text{if } y \in V_{con} \\ \mathbf{h}^{p}_{loc(y,P)} & \text{if } y \in n_P \end{cases}$$

Model Description

- **Training Objective**
  - Minimize is the negative log-likelihood of $\mathbb{D} = \{(P^i, T^i) : 1 \leq i \leq N\}$

$$J = \sum_{(P,T)\in\mathbb{D}} -\log p(T|P)$$

  - Training stage
    - The decoder generates one target token at each step, in the pre-order traversal of T
    - It ensures that the ground truth is used as the tree structure during training
    - The conditional probability

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

$$\text{prob}(\cdot|\cdot) = \frac{\exp(\text{s}(y|\mathbf{q}, \mathbf{c}, P))}{\sum_i \exp(\text{s}(y_i|\mathbf{q}, \mathbf{c}, P))}$$

◦ m denotes the size of T
◦ The goal vector & its context vector at the t-th node $\mathbf{q}_t$ $\mathbf{c}_t$

Experiment

- **Datasets: Math23K (Wang et al., 2017)**
  - 23,161 math word problems annotated with solution expressions 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
  - Problems in this dataset can be solved by one linear algebra expression
  - The solution expression can be easily transformed into the corresponding expression tree

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	

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

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

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

Experiment

- **Models for Comparison**
  - Hybrid model w/ SNI [Wang et al., 2017]
    - Combine retrieval model & seq2seq model with significant number identification(SNI)
  - Ensemble model w/ EN [Wang et al., 2018a]
    - Select the result according to models' generation probability among BiLSTM, ConvS2S, Transformer with equation normalization(EN)
  - Goal-driven tree-structured MWP solver (GTS)
    - The method proposed in this paper
  - GTS model w/o Subtree Embedding
    - To make clear the effect of subtree embedding component on the performance of GTS
    - Right sub-goal generation in the same way as the left sub-goal (that is, the subtree embedding component gets removed)

Yan Wang et al. Deep neural solver for math word problems. EMNLP. 2017
Lei Wang et al. Translating math word problem to expression tree. EMNLP. 2018.

Experiment

- **Implementation Details**
  - All the words with less than 5 occurrences are converted to a universal token "<unk>"
  - The dimensionality of word embedding layer is set to 128, all hidden states for the other layers are set to 512
  - Our model is trained for 80 epochs by Adam optimization algorithm [Kingma and Ba, 2014] where the mini-batch size is set to 64
  - The initial value of learning rate is set to 0.001, and the learning rate will be halved every 20 epochs
  - Set the dropout probability [Hinton et al., 2012] as 0.5 and weight decay as $1e-5$ to prevent overfitting
  - Last but not least, we set the beam size to 5 in beam search to generate expression trees

Experiment

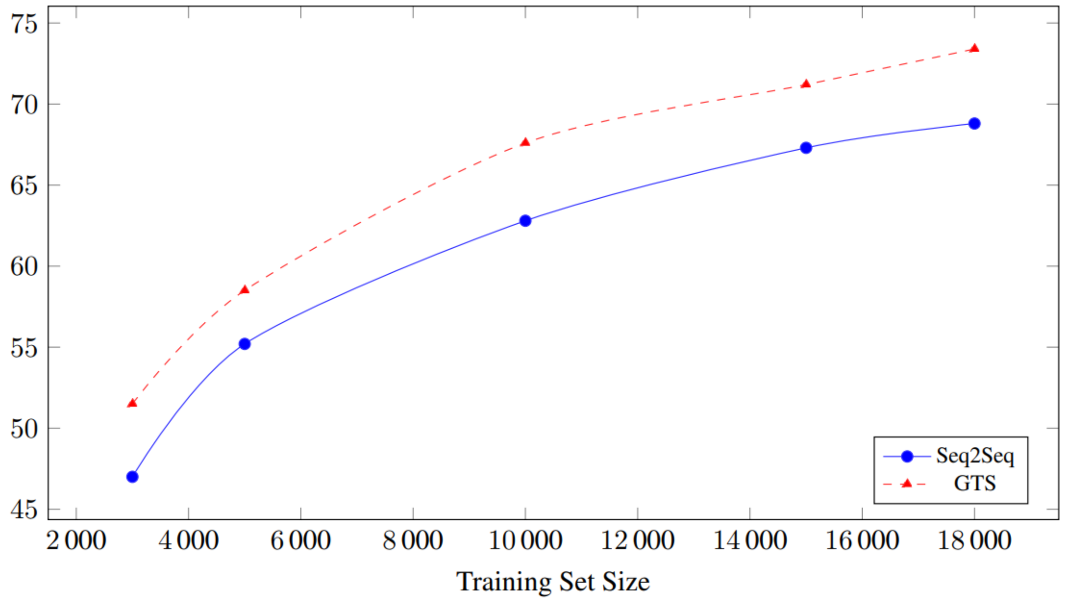
- **Results and Analyses**
  - Answer Accuracy
    - The predicted expression tree of "$n_0 + n_1$" is different from the target expression tree of "$n_1 + n_0$", but they are equivalent and their calculated values are equal
  - Goal-driven mechanism is feasible for solving the math word problems
  - The subtree embedding module is helpful and complementary to top-down goal decomposition process

Model comparison on answer accuracy

Model	Accuracy(%)
Hybrid model w/ SNI [Wang *et al.*, 2017]	64.7
Ensemble model w/ EN [Wang *et al.*, 2018a]	68.4
GTS model w/o Subtree Embedding	70.0
GTS model	**74.3**

Experiment

- **Results and Analyses**
  - Answer Accuracy vs. Training Set Size
    - Check how the performance of our model varies with respect to different numbers of training instances
  - Baseline
    - a Seq2Seq model with attention mechanism, which contains the same encoder as GTS model but takes GRU as decoder



Experiment

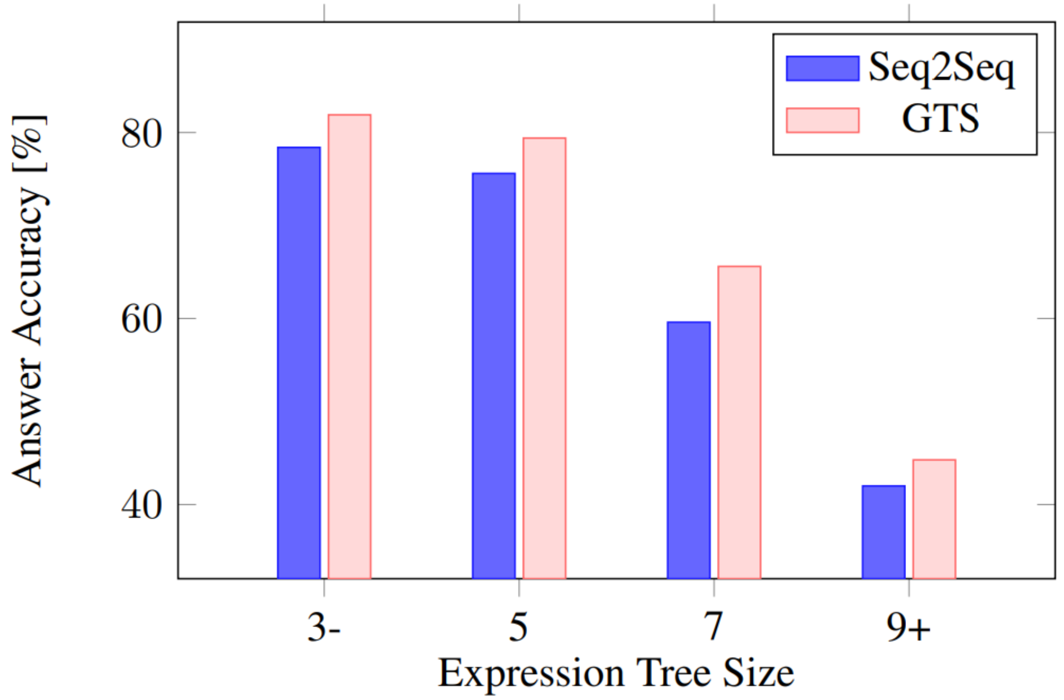
• Performance on Expression Length

- A clear tendency for answer accuracy to degrade with the growth of the problem complexity measured as the size of expression tree
- GTS model can better model the mathematical relationships of the problem in an explicit tree structure

Numbers of test instances over expression tree sizes

Expr. Tree Size	3-	5	7	9+
# Test Instances	907	2303	921	501

◦ Expression tree size is no less than 9, the tree will contain at least 4 mathematical operators



Experiment

- **Results on a Small Dataset**
  - The performance of the GTS model on another small dataset called AllArith
  - Use the same 5-fold cross validation
  - To improve the reproducibility, we repeat the experiment 20 times
  - Due to the dataset is small, perform McNemar's test and get p-value 0.001
    - It rejects the null hypothesis and this increase is statistically significant

McNemar's test

$$H_0 : p_b = p_c \qquad H_1 : p_b \neq p_c$$

$$\chi^2 = \frac{(b-c)^2}{b+c}$$

		GTS		
		GTS 1	GTS 2	..
Seq2seq	Seq2seq 1	a	b	..
	Seq2seq 2	c	d	..
	...	..	..	..

Available from https://github.com/CogComp/arithmetic

Experiment

• Case Study

- Case 1: Avoid generating mathematically invalid expressions
  - Generate the tree directly, and its sequence of pre-order traversal can be guaranteed to be computable
- Case 2: Avoid predicting spurious numbers
  - Effective size of target vocabulary is set dynamically according to the specific problem
- Case 3: The subtree embedding component
  - The subtree embedding component can prevent generating the same subtree as its left sibling when the parent node is “ + ” or “ × ”

Typical cases

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)**

Case 3: Guangming Primary School spent NUM(n_0 [288]) yuan on NUM(n_1 [12]) chairs. And then NUM(n_2 [36]) chairs of the same kind were bought. How much did the school spend on chairs?
GTS w/o Subtree Embedding: $\times \div n_0 n_1 \div n_0 n_1$;**(error)** **GTS:** $\times \div n_0 n_1 + n_1 n_2$;**(correct)**

Conclusion

- **A Goal-Driven Tree-Structured Neural Model for Math Word Problems**
  - Motivatation
    - The goal-driven mechanism in human problem solving
  - a novel neural model (called GTS) for math word problems
    - Directly predicting an expression tree
    - The information is able to flow explicitly through the expression tree by top-down goal decomposition and bottom-up subtree embedding
  - Experimental result
    - Significantly outperform previous state-of-the-art systems
  - Case study
    - Avoid generating mathematically invalid expressions and spurious numbers