Math Word Problem Solving with Explicit Numerical Values

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, Xuanjing Huang*
Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University, Shanghai, China
(qzwu17,qz,zywei,xjhuang)@fudan.edu.cn

Abstract

In recent years, math word problem solving has received considerable attention and achieved promising results, but previous methods rarely take numerical values into consideration. Most methods treat the numerical values in the problems as number symbols, and ignore the prominent role of the numerical values in solving the problem. In this paper, we propose a novel approach called NumS2T, which enhances math word problem solving performance by explicitly incorporating numerical values into a sequence-to-tree network. In addition, a numerical properties prediction mechanism is used to capture the category and comparison information of numerals and measure their importance in global expressions. Experimental results on the Math23K and APE datasets demonstrate that our model achieves better performance than existing state-of-the-art models. [1]

Problem: A school purchased several pairs of new desks and chairs for grade v_1 (15 / 1 / 3 / 7) students. Each desk is worth \$ v_2 (15 / 9.9 / 6.5), and each chair is worth \$ v_3 (10 / 8.3 / 8). The price difference between the tables and the chairs is \$ v_4 (100 / 64 / 18). There are v_5 (25 / 20% / (1/3)) more students than chairs. How many students are there?

Expression 1: v_4 / (v_2 - v_3) + v_5
Numerical expression: 100 / (15 – 10) + 25

Expression 2: v_4 / (v_2 - v_3) * (1 + v_5)
Numerical expression: 64 / (9.9 – 8.3) * (1+20%)

Expression 3: v_4 / (v_3 - v_2) * (1 + v_5)
Numerical expression: 18 / (8 – 6.5) * (1 + (1/3))

Figure 1: Example of a math word problem. The same problem with different numerical values may correspond to different math expressions. Without numerical value information, the model can hardly determine which expression is correct.

1 Introduction

Taking a math word problem as input, the math word problem solving task aims to generate a corresponding solvable expression and answer. With the advancements in natural language processing, math word problem solving has received growing attention in recent years (Roy and Roth, 2015; Mitra and Baral, 2016; Ling et al., 2017; Huang et al., 2018). Many methods have been proposed that use sequence-to-sequence (seq2seq) models with an attention mechanism (Bahdanau et al., 2014) for math word problem solving (Wang et al., 2017b, 2018b, 2019). To better utilize expression structure information, some methods use sequence-to-tree (seq2tree) models to generate expressions and have achieved promising results (Liu et al., 2019; Xie and Sun, 2019; Wu et al., 2020). These methods convert the target expression into a binary tree, and generate a pre-order traversal sequence of this expression tree based on the parent and sibling nodes of each node.

Although promising results have been achieved, previous methods rarely take numerical values into consideration, despite the fact that in math word problem solving, numerical values provide vital information. As an infinite number of numerals can appear in math word problems, it is impossible to list them all in the vocabulary. Previous methods replace all the numbers in the problems with number symbols (*e.g.*, v_1, v_2) in order in the preprocessing stage. These replaced problems are used as input

* Corresponding author.
[1] Code is available at https://github.com/qinzhuowu/NumS2T/

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 5859–5869
August 1–6, 2021.

to directly generate expressions containing number symbols. The number symbols in the expressions are then replaced with the numerical values in the original problems to obtain executable expressions. As shown in Figure 1, taking the problem with numerical values $\{v_2$=15, v_3=10, v_4=100, v_5=25$\}$ as input, the target expression of the problem would be "$v_4/(v_2 - v_3) + v_5$". However, if the number symbol $v_5 = 20\%$, the target expression for the same problem would be "$v_4/(v_2 - v_3) * (1 + v_5)$". Similarly, without numerical value information, the model can hardly determine whether the number gap between the table and the chair should be $v_2 - v_3$ or $v_3 - v_2$. As such, it will incorrectly generates the same expression for problems with different numerical values.

To address these problems, we propose a novel approach called NumS2T to better capture numerical value information and utilize numerical properties. Specifically, the proposed model uses a sequence-to-tree network with a digit-to-digit number encoder that explicitly incorporates numerical values into the model and captures number-aware problem representations. In addition, we designed a numerical properties prediction mechanism to further utilize the numerical properties. NumS2T predicts the comparative relationship between paired numerical values, determines the category of each numeral, and measures their importance for generating the final expression. With the category and comparison information, the model can better identify the interactive relationship between the numerals, and thus generate better results. With consideration of the importance of the numerals, the model can capture the global relationship between the numerals and target expressions rather than simply focusing on the local relationship between numeral pairs.

The main contributions of this paper can be summarized as follows:

- We explicitly incorporate numerical value information into math word problem solving tasks.
- We propose a numerical properties prediction mechanism to utilize numerical properties. To incorporate the local relationship between numerals and the global relationship associated with the final expression, NumS2T compares the paired numerical values, determines the category of each numeral, and then measures whether they should appear in the final expression.
- We conducted experiments on two large-scale Math23K and Ape210K datasets to verify the effectiveness of our NumS2T model. The results show that our model achieved better performance than existing state-of-the-art methods.

2 Models

In this section, we present details regarding our proposed NumS2T model. As shown in Figure 2, we use an attention-based sequence-to-tree model with a problem encoder (Section 2.2) and a tree-structured decoder to generate math expressions (Section 2.4). In addition, we explicitly incorporate numerical values to obtain number-aware problem representations (Section 2.3). Finally, we propose a numerical properties prediction mechanism to further utilize the numerical properties (Section 2.5).

2.1 Problem Definition

A math word problem $\mathrm{X} = (x_1, x_2, \ldots, x_m)$ is a sequence of m words. Our goal is to generate a math expression $\mathrm{Y} = (y_1, y_2, \ldots, y_n)$, where Y is the pre-order traversal sequence of a binary math expression tree, which can be executed to produce the answer to problem X.

Here, we replace all of the numbers in the problem X with a list of number symbols based on their order of appearance. Let $\mathrm{V_c} = (v_1, v_2, \ldots, v_K)$ be the K numbers that appear in problem X. The numerical value of the k-th number v_k is a sequence of l characters $(v_k^1, v_k^2, \ldots, v_k^l)$. The generated vocabulary $\mathrm{V_g}$ is composed of several common numbers (*e.g.*, 1,100,π) and several math operators (*e.g.*, +,-,*,/). At each time step during decoding, the NumS2T model either copies a number from V_c or generates a number from V_g.

2.2 Problem Encoder

We use a two-layer bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber, 1997) network as the encoder, which encodes the math word problem X into a sequence of hidden states

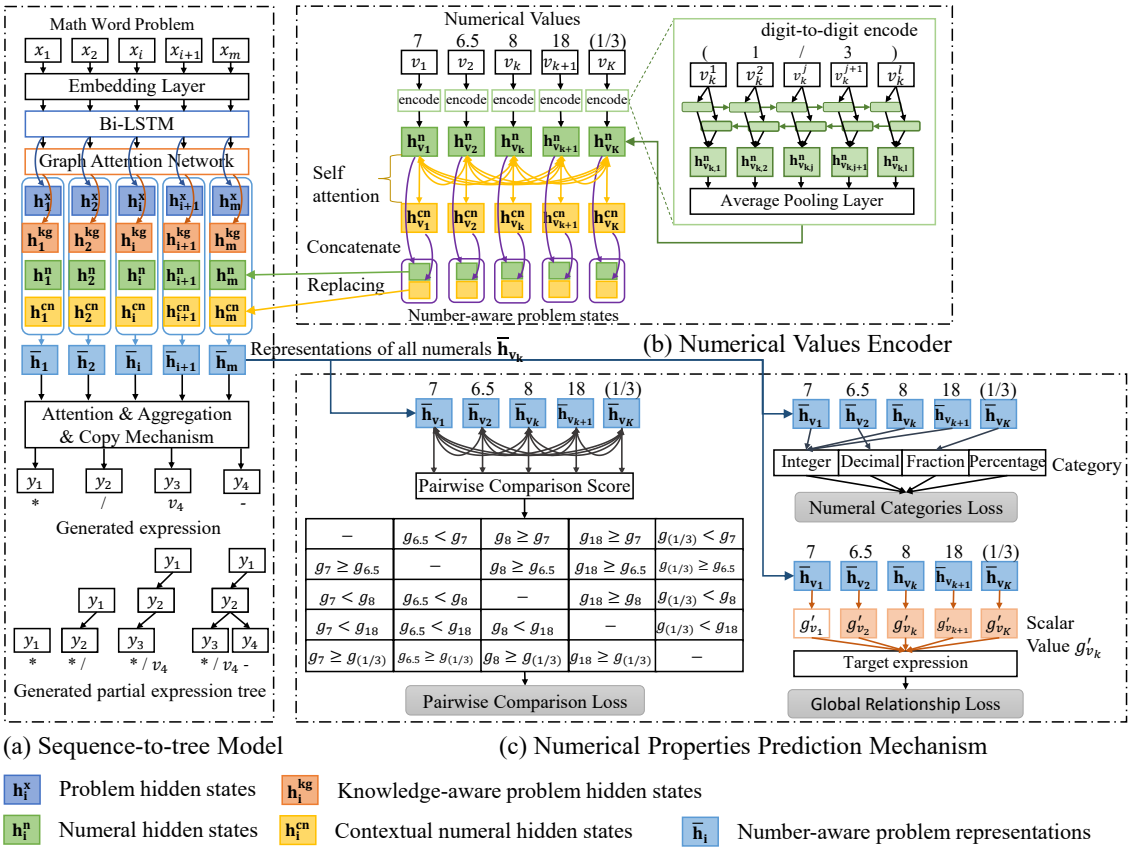

Figure 2: Main structure of our NumS2T model. Given a math word problem sequence, we use (a) an attention-based sequence-to-tree model to generate its math expression. To explicitly incorporate numerical value information, we use (b) a numerical values encoder to obtain the number-aware problem states $\mathbf{h_i^{num}}$, which are then concatenated with the problem hidden states in (a) to obtain number-aware problem representations $\overline{\mathbf{h}}_\mathbf{i}$. In addition, we propose (c) a numerical properties prediction mechanism for comparing the paired numerical values, determining the category of each numeral, and measuring whether they should appear in the target expression.

$\mathrm{H}=(\mathbf{h_1^x}, \mathbf{h_2^x}, \ldots, \mathbf{h_m^x}) \in \mathrm{R}^{m \times 2d}$ as follows:

$$\begin{aligned} \mathbf{h_i^x} &= [\overrightarrow{\mathbf{h_i^x}}, \overleftarrow{\mathbf{h_i^x}}], \\ \overrightarrow{\mathbf{h_i^x}} &= \mathrm{BiLSTM}(\mathbf{E}(x_i), \overrightarrow{\mathbf{h_{i-1}^x}}), \\ \overleftarrow{\mathbf{h_i^x}} &= \mathrm{BiLSTM}(\mathbf{E}(x_i), \overleftarrow{\mathbf{h_{i-1}^x}}). \end{aligned} \tag{1}$$

Here, word embedding vectors $\mathbf{E}(x_i)$ are obtained via a wording embedding layer $\mathbf{E}(\cdot)$. d is the dimension of the hidden state and $\mathbf{h_i^x}$ is the concatenation of the forward and backward LSTM hidden states.

Following Wu et al. (2020), we enrich the problem representations with common-sense knowledge information from external knowledge bases. The words in problem sequences X and their categories in external knowledge bases are constructed as an entity graph. In this entity graph, each word is related to its neighbor in the problem. If there are two nouns belonging to the same category in the knowledge base, these two nouns are related to their categories. See Wu et al. (2020) for more details.

The knowledge-aware problem states $\mathbf{h_i^{kg}}$ are obtained from a two-layer graph attention network (Veličković et al., 2018) on the entity graph:

$$\begin{aligned} \alpha_{ij} &= \underset{\mathrm{A_{ij}}=1}{\mathrm{softmax}}(f(\mathrm{w_h^T}[\mathrm{W_x}\mathbf{h_i^x} : \mathrm{W_x}\mathbf{h_j^x}])), \\ \mathbf{h_i^{kg}} &= \underset{t=1,\ldots,T}{||} \sigma(\sum\nolimits_{\mathrm{A_{ij}}=1} \alpha_{ij} \mathrm{W_k}\mathbf{h_j^x}), \end{aligned} \tag{2}$$

where $\mathrm{w_h^T}, \mathrm{W_x}, \mathrm{W_k}$ are weight vector and matrices. $||$ and $[:]$ are concatenation functions. $f(\cdot)$ and σ are the LeakyRelu and sigmoid activation functions. T is the number of heads in GAT layer. If the i-th word is related to the j-th word, the score of the adjacent matrix $\mathrm{A_{ij}}$ is set to 1, otherwise it is set to 0.

2.3 Number-aware Problem Representations

To solve the issues mentioned in the introduction section, we need to incorporate explicit numerical value information into NumS2T. However, there are an infinite number of numerals that can appear in math word problems. For example, among the 18,529 problems in the training set of Math23K, there are 3,058 different numerical values. Therefore, rather than list all these numerals in the vocabulary, we encode each numeral value digit by digit.

All the digits in the numerical value v_k are treated as a sequence $(v_k^1, v_k^2, \ldots, v_k^l)$ and embedded via the embed layer $\mathrm{E}(\cdot)$. Take a 5-digit value $v_k = (1/3)$ as an example, we have $\mathrm{E}(\mathrm{v_k}) \in \mathrm{R}^{5\times \mathrm{d_{emb}}}$. Similar to the architecture shown in Equation 1, we use a BiLSTM network to encode the numeral values and obtain the numeral hidden states $\mathbf{h_{v_k}}$ with an average pooling layer:

$$\begin{aligned} \mathbf{h^n_{v_{k,j}}} &= \mathrm{BiLSTM}(\mathbf{E}(v_k^j), \mathbf{h^n_{v_{k,j-1}}}), \\ \mathbf{h^n_{v_k}} &= \frac{1}{l}\sum\nolimits_{j=1}^{l} \mathbf{h^n_{v_{k,j}}}. \end{aligned} \tag{3}$$

To capture the relations and dependency between numeral pairs, we use a self-attention mechanism (Wang et al., 2017a) on the hidden state of all the numerals $\mathbf{H^n_v} = \{\mathbf{h^n_{v_k}}\}_{k=1}^{K}$ to compute the contextual numeral hidden states $\mathbf{h^{cn}_{v_k}}$:

$$\begin{aligned} \alpha_{v_k} &= \mathrm{softmax}(\ (\mathbf{H^n_v})^{\mathrm{T}} \mathrm{W_h} \mathbf{h^n_{v_k}}), \\ \mathbf{h^{cn}_{v_k}} &= \alpha_{v_k} \cdot \mathbf{H^n_v}, \end{aligned} \tag{4}$$

where α_{v_k} is the attention distribution of v_k on all the numerals in the problem X.

Combining the numeral hidden states $\mathbf{h^n_{v_k}}$, $\mathbf{h^{cn}_{v_k}}$ with the original problem hidden states $\mathbf{h^x_i}$, $\mathbf{h^{kg}_i}$, we have number-aware problem states $\mathbf{h^{num}_i}$ enhanced with explicit numeral value information:

$$\mathbf{h^{num}_i} = \begin{cases} [\mathbf{h^n_{v_k}} : \mathbf{h^{cn}_{v_k}}] & x_i = v_k \\ [\mathbf{h^x_i} : \mathbf{h^{kg}_i}] & x_i \text{ is not a number} \end{cases} \tag{5}$$

The final number-aware problem representations are obtained by concatenating the problem hidden states $\mathbf{h^x_i}$, the knowledge-aware problem states $\mathbf{h^{kg}_i}$ and the number-aware problem states $\mathbf{h^{num}_i}$:

$$\mathbf{\overline{h}_i} = [\mathbf{h^x_i} : \mathbf{h^{kg}_i} : \mathbf{h^{num}_i}]. \tag{6}$$

2.4 Tree Structured Decoder

Previous works (Xie and Sun, 2019; Liu et al., 2019; Wu et al., 2020) have confirmed that a sequence-to-tree model can better represent the expression structures than a sequence-to-sequence model, because a tree structured decoder can capture the global expression information and focus on the features of adjacent nodes.

The tree structured decoder takes the final number-aware problem representations $\mathbf{\overline{h}_i}$ as input and generates the target expression from top to bottom. The target expression can be regarded as a pre-order traversal of a binary tree, with operators as internal nodes and numbers as leaf nodes. The decoder is a one-layer LSTM, which updates its states as follows:

$$\mathbf{s_{t+1}} = \mathrm{LSTM}([\mathbf{E}(y_t) : \mathbf{c_t} : \mathbf{r_t}], \mathbf{s_t}). \tag{7}$$

At time step t+1, the decoder uses the last generated word embedding $\mathbf{E}(y_t)$, the problem context state $\mathbf{c_t}$ and the expression context state $\mathbf{r_t}$ to update its previous hidden state $\mathbf{s_t}$.

The problem context state $\mathbf{c_t}$ is computed via attention mechanism as follows:

$$\begin{aligned} \alpha_{ti} &= \mathrm{softmax}(\tanh(\mathrm{W_h}\mathbf{\overline{h}_i} + \mathrm{W_s}[\mathbf{s_t} : \mathbf{r_t}])), \\ \mathbf{c_t} &= \sum_{i=1}^{m} \alpha_{ti} \mathbf{\overline{h}_i}, \end{aligned} \tag{8}$$

where $\mathrm{W_h}$, $\mathrm{W_s}$ are weight matrices. α_{ti} is the attention distribution on the number-aware problem representations $\mathbf{\overline{h}_i}$.

The expression context state $\mathbf{r_t}$ is computed via a state aggregation mechanism (Wu et al., 2020). It describes the global representation of the partial expressions $y_{<t} = (y_1, y_2, \ldots, y_{t-1})$ being generated by the decoder. At time step t, the decoder aggregates each node's context state with its neighbor nodes in the generated partial expression tree. The aggregation functions are as follows:

$$\begin{aligned} \mathbf{r_t^0} &= \mathbf{s_t}, \\ \mathbf{r_t^{\eta+1}} &= \sigma(\mathrm{W_r}[\mathbf{r_t^\eta} : \mathbf{r^\eta_{t,p}} : \mathbf{r^\eta_{t,l}} : \mathbf{r^\eta_{t,r}}]), \end{aligned} \tag{9}$$

where σ is the sigmoid function and $\mathrm{W_r}$ is a weight matrix. $\mathbf{r_t^0}$ is initialized with decoder hidden state $\mathbf{s_t}$ when $\eta = 0$,. $\mathbf{r_{t,p}}, \mathbf{r_{t,l}}, \mathbf{r_{t,r}}$ are the context state of the parent node, the left child node, and the right child node of y_t in the expression tree. $\mathbf{r_t^{\eta+1}}$ represents the expression context state updated with

global information from all nodes in the generated partial expression.

Lastly, the decoder can generate a word from a given vocabulary $\mathrm{V_g}$. It can also generate a number symbol in $\mathrm{V_c}$, and use it to copy a number from the problem X. The final distribution is the combination of the generated probability and copy probability:

$$\begin{aligned}
\overline{\mathbf{H}}_{\mathbf{v}} &= \{\overline{\mathbf{h}}_{\mathbf{v_k}}\}_{k=1}^{K}, \\
p_c &= \sigma(\mathrm{W_z}[\mathbf{s_t} : \mathbf{c_t} : \mathbf{r_t}] + \mathrm{W_v}\overline{\mathbf{H}}_{\mathbf{v}}), \\
\mathrm{P_c}(y_t) &= \mathrm{softmax}(f([\mathbf{s_t} : \mathbf{c_t} : \mathbf{r_t} : \overline{\mathbf{H}}_{\mathbf{v}}])), \\
\mathrm{P_g}(y_t) &= \mathrm{softmax}(f([\mathbf{s_t} : \mathbf{c_t} : \mathbf{r_t}])), \\
\mathrm{P}(y_t|y_{<t},\mathrm{X}) &= p_c\mathrm{P_c}(y_t) + (1-p_c)\mathrm{P_g}(y_t).
\end{aligned} \tag{10}$$

Here, $\overline{\mathbf{H}}_{\mathbf{v}}$ are the number-aware problem representations of all the numerals v_k in X. $\mathrm{W_z}, \mathrm{W_v}$ are the weight matrices. $f(\cdot)$ is a perception layer. p_c is the probability that the current word is a number copied from the problem.

2.5 Numerical Properties Prediction Mechanism

Our NumS2T model explicitly incorporates numerical values information. Furthermore, utilize the numerical properties to the degree possible through a numerical properties prediction mechanism. We consider three numerical properties to be useful for solving math word problems:

Pairwise Numeral Comparison. If we consider the question "What is the difference between v_1 and v_2," the comparative relationship between these two numerals can help the model decide whether to generate $v_1 - v_2$ or $v_2 - v_1$. In this paper, we compare each numeral v_k in the question with the other numerals. Then, we calculate the pairwise comparison scores z_{kj} based on their number-aware problem representations, and we optimize the pairwise comparison loss to assign numerals with larger numerical values higher pairwise comparison scores. The pairwise comparison loss $\mathcal{L}_{CR}$ is calculated as follows:

$$\begin{aligned}
g_{v_k} &= \sigma(\mathrm{W_h}\overline{\mathbf{h}}_{\mathbf{v_k}}), \\
z_{kj} &= \begin{cases} max(0, g_{v_j} - g_{v_k}) & \text{if } v_k \geq v_j \\ max(0, g_{v_k} - g_{v_j}) & \text{if } v_k < v_j \end{cases}, \\
\mathcal{L}_{CR} &= -\frac{1}{K^2}\sum_{k=1}^{K}\sum_{j=1}^{K} z_{kj},
\end{aligned} \tag{11}$$

Numeral categories. In the sentence "the number of apples is 5 more than the number of pears," replacing the numeral 5 with the integer 100 may not affect the structure of the target expression, but replacing the numeral 5 with 20% may change the structure from "+5" to "*(1 + 20%)". We roughly divide all numbers into four categories: {integer, decimal, fraction, percentage}, and assign a category label C = {1,2,3,4}, respectively. Given the number-aware problem representations $\overline{\mathbf{h}}_{\mathbf{v_k}}$ for each numeral v_k, we calculate the category score distribution $\mathrm{P}(C_{v_k}|\overline{\mathbf{h}}_{\mathbf{v_k}})$ and then minimize the negative log likelihood:

$$\begin{aligned}
\mathrm{P}(C_{v_k}|\overline{\mathbf{h}}_{\mathbf{v_k}}) &= \mathrm{softmax}(\mathrm{W_c}\overline{\mathbf{h}}_{\mathbf{v_k}}), \\
\mathcal{L}_{CA} &= -\frac{1}{K}\sum_{k=1}^{K}\log \mathrm{P}(C_{v_k}|\overline{\mathbf{h}}_{\mathbf{v_k}}).
\end{aligned} \tag{12}$$

Global relationship with target expressions. Current models tend to focus on the local relationship between numerals, while sometimes these numerals are not related to the target expression. Given "3 bags of rice weighing 60 kg," the numeral 3 is highly correlated with 60. However, if the problem relates to the total price of the rice rather than the weight of each bag of rice, the numeral 3 is not so important for generating the target expression. The NumS2T model predicts a scalar value g'_{v_k} for each numeral that denotes whether this numeral will be used in a math expression. The importance label a_{v_k}=1 when v_k is used in the ground truth math expression, otherwise a_{v_k}=0. The supervised loss is defined by:

$$\begin{aligned}
g'_{v_k} &= \sigma(\mathrm{W_g}\overline{\mathbf{h}}_{\mathbf{v_k}}), \\
\mathcal{L}_{GR} &= -\frac{1}{K}\sum_{k=1}^{K} a_i \log g'_{v_k} + (1-a_i)\log(1-g'_{v_k}).
\end{aligned} \tag{13}$$

2.6 Training

During training, for each question–expression pair (X, Y), we first train the NumS2T by optimizing the maximum likelihood estimation (MLE) loss $\mathcal{L}_l$ on the probability distribution $\mathbf{P}(y_t|y_{<t}, \mathrm{X}))$. Then, the final loss function $\mathcal{L}$ is a combination of the MLE loss and three numerical properties loss functions:

$$\begin{aligned}
\mathcal{L}_l &= -\frac{1}{n}\sum_{i=1}^{n}\log \mathbf{P}(y_t|y_{<t}, \mathrm{X})), \\
\mathcal{L} &= \mathcal{L}_l + \beta_1\mathcal{L}_{CR} + \beta_2\mathcal{L}_{CA} + \beta_3\mathcal{L}_{GR}.
\end{aligned} \tag{14}$$

Here, $\beta_1, \beta_2, \beta_3$ are hyper-parameters.

3 Experiment

3.1 Dataset

We present the experimental results of math word problem solving using our proposed models on the Math23K (Wang et al., 2017b) and Ape210K (Zhao et al., 2020)[2] datasets. Following Xie and Sun (2019), we removed the problems that the corresponding expressions could not be executed to obtain the given answers and the problems that omit intermediate calculation expressions. For Math23K, following previous studies (Xie and Sun, 2019; Wu et al., 2020), we randomly split the dataset into a training set, a development set and a test set with 18,529, 2,316, 2,316 problems. For Ape210K, we use the official data partition. There are 166,270, 4,157, and 4,159 problems in our training set, development set and test set, respectively.

We report answer accuracy as the main evaluation metrics of the math word problem solving task.

3.2 Implementation Details

In this paper, we truncate the problem to a max sequence length of 150, and the expression to a max sequence length of 50. We select 4,000 words that appear most frequently in the training set of each dataset as the vocabulary, and replace the remaining words with a special token UNK. We initialize the word embedding with the pre-trained 300-dimension word vectors[3]. The problem encoder used two external knowledge bases: Cilin (Mei, 1985) and Hownet (Dong et al., 2010). The number of heads T in GAT is 8. The hidden size is 512 and the batch size is 64. We use the Adam optimizer (Kingma and Ba, 2014) to optimize the models an the learning rate is 0.001. We compute the final loss function with $\beta_1, \beta_2, \beta_3$ of 0.5. Dropout (Srivastava et al., 2014) is set to 0.5. Models are trained in 80 epoches for the Math23K dataset and 50 epoches for the Ape210K dataset. During testing, the beam size is set to 5. Once all internal nodes in the expression tree have two child nodes, the decoder stops generating the next word. The hyper-parameters are tuned on the valid set.

[2]https://github.com/yuantiku/ape210k

[3]https://github.com/Embedding/Chinese-Word-Vectors

Models	Math23K	APE210K
DNS	58.1%	-
DNS-Retrieval	64.7%	-
S2S	66.7%	56.6%
RecursiveNN	68.7%	-
Tree-Decoder	69.0%	66.5%
GTS	74.3%	67.7%
KA-S2T	76.3%	68.7%
NumS2T	**78.1%**	**70.5%**

Table 1: Answer accuracy of our model and other state-of-the-art models on the Math23K and APE210K datasets.

3.3 Baselines

We compare our proposed NumS2T model with the following baseline models: **DNS** (Wang et al., 2017b) is a seq2seq model with a two-layer GRU as an encoder and a two-layer LSTM as a decoder. **DNS-Retrieval** is a variant of DNS that combines a retrieval model. **S2S** (Wang et al., 2018a) is a standard bidirectional LSTM-based seq2seq model with an attention mechanism. **RecursiveNN** (Wang et al., 2019) uses a recursive neural network on the predicted tree structure templates **Tree-Decoder** (Liu et al., 2019) is a seq2tree model with a tree structured decoder. The decoder generates each node based on its parent node and its sibling node. **GTS** (Xie and Sun, 2019) generates each node based on its parent node and its left sibling subtree embedding. The subtree embedding is obtained by merging the embedding of the subtree from bottom to top. **KA-S2T** (Wu et al., 2020) is a seq2tree model with external knowledge and a state aggregation mechanism. The decoder use a two-layer GCN to recursively aggregate neighbors of each node in the partial expression tree.

3.4 Results Analysis

The main evaluation results are presented in Table 1. Compared with baseline methods, our model obtains the highest answer accuracy of 78.1% in the Math23K dataset and 70.5% in the APE210K dataset, which is significantly better than other state-of-the-art methods. The experimental results provide the following observations:

1) The methods with a tree-structured decoder (Tree-Decoder, GTS, KA-S2T) perform better than methods with a sequence-structured decoder (DNS, S2S). These methods treat the math expression as a binary tree and directly use adjacent nodes in the

tree instead of the previous word in the sequence to generate the next word. In this way, the model can better capture the structure information of the math expressions.

2) The KAS2T model with external knowledge performs better than GTS, which proves that external knowledge enables the model to obtain better interaction between words.

3) NumS2T outperforms all the other baselines. This result shows the effectiveness of the explicitly incorporated numerical values and use of a numerical properties prediction mechanism.

3.5 Ablation Study

Effect of explicitly incorporating numerical values: We designed several NumS2T variants that reduce the numerical values incorporated in the model. Here, "NumS2T w/o Numerals" means that we remove the character-level numeric value encoder. An input example is "Alan bought v_1 apples for \$ v_2". "NumS2T w/o Symbols" means that we not only remove the character-level numeric value encoder, but also replace the math symbols in math problems with character-level numeric values. An input example is "Alan bought 2 5 apples for \$ 1 5 0".

Table 2 shows the results of these different variants, from which we can see:

1)The experimental results show that model performance of "NumS2T w/o Symbols" is significantly reduced in both datasets. We believe this is because directly replacing the number symbols will make it difficult for the model to obtain the overall representation of each number.

2) The use of a self-attention mechanism significantly improves the accuracy by 0.8% in Math23K and 0.7% in APE210K. This is because the same numerical value may describe different information in different problems. Therefore, the self-attention mechanism combines numerical values with other numerical values in the problem, which helps to model numerical information and the relations between these numerals.

3) Without numerical values, the answer accuracy of "NumS2T w/o Numerals" would be reduced to 76.6% and 69.2%. The results show the benefit of explicitly incorporating numerical values.

Effect of the numerical properties prediction mechanism: Table 3 shows the results of several NumS2T variants designed to measure the effect of the numerical properties prediction mechanism. From the table we can observe that:

Models	Math23K	APE210K
KA-S2T	76.3%	68.7%
NumS2T w/o Symbols	75.4%	64.4%
NumS2T w/o Numerals	76.6%	69.2%
NumS2T w/o SelfAtt	77.3%	69.8%
NumS2T	**78.1%**	**70.5%**

Table 2: Ablation study on reducing the numerical values incorporated into the model.

Models	Math23K	APE210K
KA-S2T	76.3%	68.7%
NumS2T-base	77.0%	69.6%
NumS2T-base + CR	77.7%	70.1%
NumS2T-base + CA	77.4%	70.0%
NumS2T-base + GR	77.3%	69.8%
NumS2T	**78.1%**	**70.5%**

Table 3: Ablation study on reducing the numerical properties used in the numerical properties prediction mechanism. CR, CA and GR respectively indicate pairwise numeral comparison, numeral category and global relationship with the target expression.

1) NumS2T-base is the variant of NumS2T without the numerical properties prediction mechanism. Without numerical properties, the answer accuracy in the Math23K and APE210K datasets are reduced to 77.0% and 69.6%, which show that the numerical properties prediction mechanism contributes considerably to improving performance. In addition, NumS2T-base still outperforms the state-of-the-art baseline KA-S2T, which once again proves the effectiveness of explicitly incorporating numerical values.

2) The use of pairwise numeral comparison, numeral category and global relationship with a target expression can improve accuracy by approximately 0.6%, 0.4% and 0.3%, respectively. Their combination achieves further improvements in model performance. These results show the effectiveness of the numerical properties prediction mechanism because it enables the model to further utilize numerical properties.

Model performance on problems with a different number of numerals: Table 4 shows the results for how accuracy changes as the number of numerals in the problem increases. The NumS2T model outperforms the best-performing baseline with respect to problems with a different number of

Math23K				
Num.	Prop.	KA-S2T	NumS2T	Imp.(↑)
≤1	2.0%	80.9%	83.0%	2.1%
2	36.8%	84.6%	85.1%	0.5%
3	46.1%	77.4%	78.4%	1.0%
4	11.4%	58.3%	60.6%	2.3%
5	2.8%	45.2%	54.9%	9.7%
6	0.7%	33.3%	46.7%	13.4%
≥ 7	0.3%	12.5%	37.5%	25.0%
APE210K				
Num.	Prop.	KA-S2T	NumS2T	Imp.(↑)
≤1	9.1%	67.9%	71.4%	3.5%
2	34.4%	74.6%	75.5%	0.9%
3	36.9%	72.2%	75.6%	3.4%
4	12.7%	53.2%	57.4%	4.2%
5	3.6%	30.1%	43.7%	4.6%
6	1.4%	40.7%	54.2%	13.5%
≥ 7	1.9%	19.0%	27.9%	8.9%

Table 4: Model performance on problems with a different number of numerals. Prop. denotes the proportion of these problems in the dataset. Imp. denotes the accuracy improvement between NumS2T and KA-S2T.

Problem:	Each football is worth $ 76, and each basketball is worth $ 45. The school bought the same number of basketballs and footballs, with a price difference of $ 248. How many footballs did the school buy?
KA-S2T:	248/(45-76)
NumS2T:	248/(76-45)
Problem:	There are 250 pear trees in the orchard, 25% more than peach trees. There are 3 times as many orange trees as pear trees. How many more orange trees are there than peach trees?
KA-S2T:	(250*3)-(250-25%)
NumS2T:	(250*3)-250/(1-25%)
Problem:	The concert was held in a hall with 80 seats. 52 tickets have been sold, each priced at $ 25. How much is the ticket revenue?
KA-S2T:	(80-52)*25
NumS2T:	52*25

Table 5: Three cases of generated expressions by KA-S2T (Wu et al., 2020) and NumS2T.

numerals. In addition, as the number of numerals in the problems increase, the performance gap between NumS2T and KAS2T also increases. This is because with more numerals in the problem, NumS2T, which explicitly incorporate numerical value information, is able to more readily achieve better performance. Meanwhile, NumS2T also achieved a considerable improvement on problems with only one numeral. This further demonstrates the effect of utilizing numerical category information and global relationship information.

3.6 Case Study

Table 5 shows three cases generated by KA-S2T (Wu et al., 2020) and our NumS2T model. In the first problem, without numerical values, KA-S2T incorrectly uses the smaller value to subtract the larger value when calculating the price difference between footballs and basketballs. This case requires the model to choose the larger value between two numerals. Our NumS2T model can better handle this problem. In the second problem, KA-S2T replaces all of the numerals in the problems with number symbols (v_1, v_2) and does not know that v_2=20% is not an integer. Our proposed method can capture numerical values and numeral category information to generate correct results. In the third problem, 80 seats and 52 tickets are strongly semantically related, so KA-S2T generates the sub-expression "80-52". However, this problem is about the fares that have already been sold rather than how many tickets are left. With numerical properties, NumS2T is able to realize that 80 is not related to the target expression and should not appear in the generated result.

4 Related Work

Math Word Problem Solving: In recent years, Seq2Seq (Sutskever et al., 2014) has been widely used in math word problem solving tasks (Ling et al., 2017; Wang et al., 2017b, 2018a). To better utilize expression structure information, recent studies have used Seq2Tree models (Liu et al., 2019; Zhang et al., 2020a). Xie and Sun (2019) proposed a tree structured decoder that uses a goal-driven approach to generate expression trees. Wu et al. (2020) proposed a knowledge-aware Seq2Tree model with a state aggregation mechanism that incorporates common-sense knowledge from external knowledge bases. Recently, several methods have attempted to use the contextual information of the numbers in the problem. Li et al. (2019) propose a group attention mechanism to extract quantity-related features and quantity-pair features. Zhang et al. (2020b) connects each

number in the problem with nearby nouns to enrich the problem representations.

However, these methods rarely take numerical values into consideration. They replace all the numbers in the problems with number symbols and ignore the vital information provided by the numerical values in math word problem solving. As such, these methods will incorrectly generates the same expression for problems with different numerical values.

Numerical Value Representations: Some recent studies have explored the numerical value representations in language models (Naik et al., 2019; Chen et al., 2019; Wallace et al., 2019). Spithourakis and Riedel (2018) investigated several of the strategies used for language models for their possible application to model numerals. Gong et al. (2020) proposed the use of contextual numerical value representations to enhance neural content planning by helping models to understand data values. To incorporate numerical value information into math word solving tasks, we use a digit-to-digit numerical value encoder to obtain the number-aware problem representations. To further utilize the numerical properties, we propose a numerical properties prediction mechanism.

5 Conclusion

In this study, we proposed a novel approach called NumS2T, that better captures numerical value information and utilizes numerical properties. In this model, we use a digit-to-digit numerical value encoder to explicitly incorporate numerical values. In addition, we designed a numerical properties prediction mechanism that compares the paired numerical values, determines the category of each numeral, and measures whether they should appear in the final expression. Experimental results show that our proposed NumS2T model outperforms other state-of-the-art baseline methods.

Acknowledgments

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No. 2018YFB1005100), National Natural Science Foundation of China (No. 62076069, 61976056), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Numeracy-600K: Learning numeracy for detecting exaggerated information in market comments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6307–6313, Florence, Italy. Association for Computational Linguistics.

Zhendong Dong, Qiang Dong, and Changling Hao. 2010. HowNet and its computation of meaning. In *Coling 2010: Demonstrations*, pages 53–56, Beijing, China. Coling 2010 Organizing Committee.

Heng Gong, Wei Bi, Xiaocheng Feng, Bing Qin, Xiaojiang Liu, and Ting Liu. 2020. Enhancing content planning for table-to-text generation with data understanding and verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2905–2914, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 419–428, Melbourne, Australia. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167, Florence, Italy. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural*

Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2370–2379, Hong Kong, China. Association for Computational Linguistics.

Jiaju Mei. 1985. *Tongyi ci cilin*. Shangai cishu chubanshe.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, Berlin, Germany. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380, Florence, Italy. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, Melbourne, Australia. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069, Brussels, Belgium. Association for Computational Linguistics.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Shen. 2019. Template-based math word problem solvers with recursive neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:7144–7151.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017a. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017b. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2020. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7137–7146, Online. Association for Computational Linguistics.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. 2020a. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4011–4017. International Joint Conferences on Artificial Intelligence Organization. Main track.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages

3928–3937, Online. Association for Computational Linguistics.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*.