Math Word Problem Solving with Dual Solving Thinking Structure

He Wang
School of Artificial Intelligence
South China Normal University
Guangzhou, China
1437393259@m.scnu.edu.cn

2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT) | 979-8-3503-8555-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/AINIT61980.2024.10581632

***Abstract*—Math Word Problem, abbreviated as MWP, constitute a class of elementary mathematical application scenarios, commonly encountered within primary secondary educational curricula. Such problems are characterized by their presentation in natural language, eschewing the utilization of intricate mathematical symbols or formulas. The focal task of model geared towards addressing Math Word Problem is twofold: first, to discern and interpret the mathematical concepts encapsulated within the textual descriptions provided, and second, to formulate an appropriate mathematical expression or equation reflective of the problem's essence. Subsequently, these models proceed to compute the solution corresponding to the derived expression, thereby completing the problem-solving process. Currently, advancements in this domain primarily revolve around improvements to encoders and decoders within models, with a relatively narrow scope of thinking processes during model solving. Consequently, the efficacy of expression solving for complex semantic Math Word Problem remains to be enhanced. In contrast to the prevalent sequence-to-tree approach, humans often employ a top-down thinking process to solve Math Word Problem. We introduce a Math Word Problem solving model featuring a dual-thinking structure, comprising a top-down solver, a step-by-step anthropomorphic thinking solver, and a consistent comparison learning module. By aligning expressions derived from two distinct solvers at multiple granularities to maintain consistency, we aim to enhance reasoning correctness. Experimental results across multiple datasets demonstrate the superiority of our approach over existing baselines, particularly in achieving notable improvements in solving accuracy for complex problems.**

Keywords-Math Word Problem, artificial intelligence, top-down solver, step-by-step anthropomorphic thinking solver, dual solving thinking structure

I. INTRODUCTION

Math Word Problem denote a class of elementary and middle school-level mathematical application problems characterized by their absence of intricate formulaic notation. The computational model for solving Math Word Problem endeavors to derive an expression reflective of the provided textual context within the problem and subsequently ascertain the corresponding solution. Given the absence of convoluted mathematical symbols in Math Word Problem, the emphasis within the mathematical application problem-solving model lies in adeptly capturing the semantic associations between the textual elements of the problem and the numerical values embedded therein. Automated solving of Math Word Problem is a significant subfield within the domain of artificial intelligence reasoning, Because the performance of such models in automated problem-solving serves as a pivotal metric for evaluating the level of artificial intelligence reasoning prowess [1]. Broadly speaking, the automatic solver for Math Word Problem is called the automatic solving model of Math Word Problem. Its purpose is to let the machine read and understand the mathematical problems described in natural language, and then deduce the solution of the mathematical problems by extracting the key words and values in the stem of the mathematical problems. In this process, the mathematical logic related to problem solving needs to be understood by machines, and machine learning and the use of relevant mathematical logic knowledge are also needed to judge and solve similar types of Math Word Problem more efficiently. More specifically, the solver in automatic solving model solver for Math Word Problem must generate a solution based on the given problem description and mathematical prior knowledge, and formulate an expression that conforms to the industry norms and can correctly calculate and solve the problem. This task requires the machine to have a deeper understanding of the natural language text, and requires the solver in the solving model to make better use of the key information in the natural language text to reason and generate a correct and interpretable solving process. From the early symbolic reasoning to probabilistic reasoning, and then to the current neural symbolic reasoning stage, the reasoning rules also reflect the rapid development of the automatic solution of mathematical problems in the field of artificial reasoning [2].

In recent years, many models still use tree-based neural models to generate mathematical application expressions to solve problems, which are more satisfactory than before, but because the text information, the relationship between numbers, and the different context combinations of words and numbers in mathematical problems will have an impact on the results of solving problems. At the same time, many tree-based neural network solving models still use the traditional text feature extraction module [3], which is far from meeting the needs of automatic solving of mathematical word problems, and needs to update the solving method. Therefore, we have designed a mathematical word problem solving model based on dual-solving thinking that considers numerical properties and mathematical formula exchange laws. The main contributions of our approach are as follows:

- We introduced a numerical feature prediction mechanism into the top-down solver based on the

seq2tree model, enabling the machine to better understand the numerical properties in the problem statements of Math Word Problem, thereby improving the accuracy of expressions generated during the solving process by the improved seq2tree model.

- We introduced a step-by-step anthropomorphic thinking solver, making the entire solving model composed of two solvers, thus allowing the solving model to have two solving approaches for the same Math Word Problem.
- We used a consistent comparison learning module to align and learn the expression reasoning processes under the two solvers.
- Experimental results on multiple standard datasets show that our solving model outperforms many baselines.

II. Literature Review

In fact, at present, the concept of automatic solution of math problems has gone through three stages.

The first stage is the initial stage from the 1970s to the 1990s, that is, the stage based on symbolic reasoning. At this time, most of the work of machine automatic solving is artificial rules and pattern matching, which relies heavily on artificial interpretation and definition [4].

The second stage is from 2011 to 2017, which is based on probability and statistical reasoning. With the help of semantic analysis and other tools, the sentences in the problem can be mapped into some structural logic expressions, which effectively strengthens the reasoning effect. During this period, due to the development of machine learning, some statistical learning methods have also been applied by researchers in the field of automatic Math Word Problem solving. First, the sub-problems of Math Word Problem solving are designed, and then the Math Word Problem solving is completed by solving the sub-problems. At this stage, solution models based on templates and tree structures have been proposed one after another [5].

The third stage is from 2018 to now, that is, the stage of reasoning based on neural symbols. With the help of more perfect and cutting-edge deep learning technology, the field of automatic solution of Math Word Problem has greatly improved the accuracy of automatic solution of mathematical problems in primary and junior middle schools. There are two main improvement directions in this stage: one is to fine-tune semantic analysis technology; The second is to try to combine in-depth learning with large language models [6]. The following research mainly focuses on using the advantages of deep representation learning method in neural symbolic reasoning model and combining with large-scale data sets to enhance the performance of solving model and improve the accuracy of solving.

We have concurrently considered both the extraction of characteristics from mathematical problems and the beneficial impact of integrating Pre-trained Language Models (PLMs) on model performance. We present a Math Word Problem solving model with a dual-cognitive architecture, which adeptly amalgamates two solving modules possessing distinct cognitive advantages. The figure 1 of the overall structure is as follows:

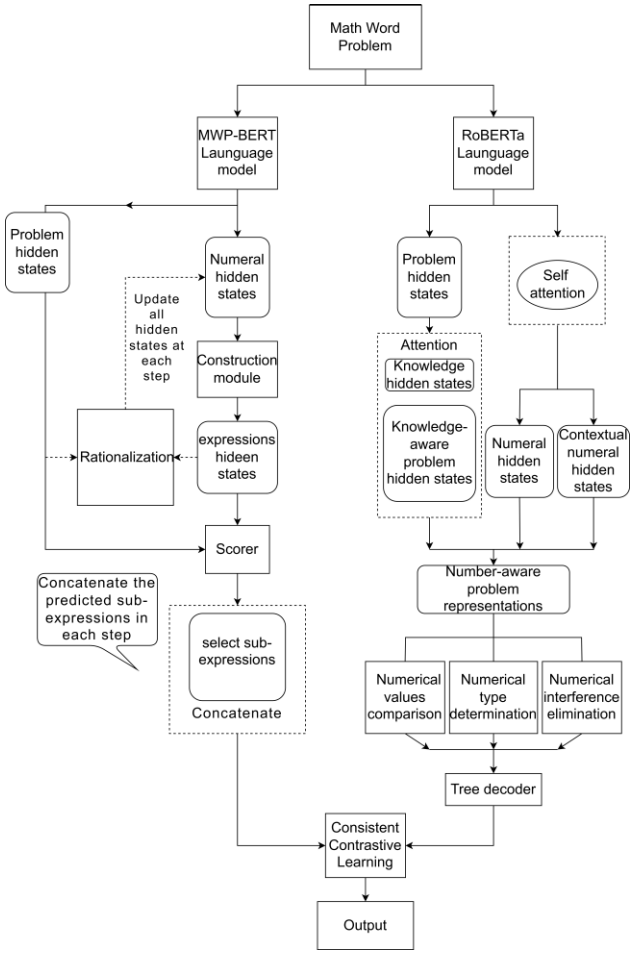

Figure 1. Structure of DTS.

III. Top-down Solver

The "double thinking" in our model is mainly realized by two different solvers, of which the top-down solver is essentially a derivative of the sequence-to-tree model, which is based on the tree decoder in the KA-S2T model. Three numerical property modules are designed and added to enhance the accuracy of the tree solver to generate the target expression. In the top-down solver, we use the widely recognized pre-trained language model RoBERTa as the encoder [7].

A. Definition of the Form of the Question

We represent a mathematical problem P containing m words by the sequence $X = (x_1, x_2, \ldots, x_m)$, and our goal is to generate the mathematical expression $Y= (y_1, y_2, \ldots, y_n)$, Y Table Preorder traversal sequence of a mathematical expression tree. We replace all the numbers in the problem X by a list of other number symbols in the order in which they appear, and let $V_c = (v_1, v_2, \ldots, v_k)$ is the number of K occurrences in question P, using l characters $(v_k^1, v_k^2, \ldots, v_k^l)$ represents the value of the kth number v_K in the problem. The generate vocabulary V_g consists of a numb of common numbers and mathematical operators. At each time step during decoding, solving the model top-down requires either copying a number from V_c or generating a number from V_g.

B. Top-Down Solver Structure

The top-down solver takes a Sequence-to-Tree model as a core, and is assisted by a numerical value type module, a numerical value size comparison module and a numerical value interference elimination module to enhance the solving and deducing effect; the expression deducing process is the same as that of the mainstream Sequence-to-Tree model, And finally generate a target expression, wherein that target expression can be regarded as a pre-order traversal of a binary tree, an operator is use as a vertex node, and a number and the operator are used as leaf nodes [8].

We use a large-scale pre-trained language model RoBERTa, by which problem P is encoded as a hidden state representation $H=\{h_1, \cdots, h_N\} \in R^{N\times 2d}$, where N = m + l, m is the number of words in problem P, l is the numerical values in problem P, and 2D is the dimension of the hidden vector. At the same time, we introduce the external knowledge base to enrich the representation of the problem [9]. The idea is that the words in the sequence X of the problem P and the categories in the external knowledge base are constructed as an entity graph, and each word in the entity graph is associated with the adjacent words in the problem, that is, if two nouns in the knowledge base belong to the same category, they are related categories, and the knowledge perception problem vector h_i^{kn} Generated by the graph attention network:

$$h_i^{kn} = ||_{t=1,\ldots,T}\, \sigma\left(\sum_{A_{ij}=1} \alpha_{ij}\, W_k\, h_j^p\right) \tag{1}$$

In the above formula h_i^p Table problem hidden state, Wx, Wk is the weight vector and matrix, | | is the link function, σ is the sigmoid activation function, T is the number of attention heads of GAT, if the ith word is not related to the jth word, then the adjacency matrix A_{ij} score is 0, if the ith word is related to the jth word, Then the adjacency matrix A_{ij} scores 1.

We encode all numerical sequences in the numerical value $V_k(v_k^1, v_k^2, \ldots, v_k^l)$ Embedding through the embedding layer E (·), taking a 5-bit value $V_k = (1/5)$ as an example, first encoding the value through a large-scale language model to obtain a digital hidden vector$h_{v_k}^n$:

$$h_{v_{k,j}}^n = Roberta\left(E\left(v_k^j\right), h_{v_{k,j-1}}^n\right) \tag{2}$$

$$h_{v_k}^n = \frac{1}{l}\sum_{j=1}^{l} h_{v_{k,j}}^n \tag{3}$$

We use the self-attention mechanism to capture the relationships and dependencies between pairs of numbers, that is, we use the attention mechanism to contextually hide the hidden state of all numbers. $h_{v_k}^c$Calculation of:

$$\alpha_{v_k} = softmax\left((H_v^n)^T W_h h_{v_k}^n\right) \tag{4}$$

$$h_{v_k}^c = \alpha_{v_k} \cdot H_v^n \tag{5}$$

Among them, h_{v_k} is the attention distribution of v_k on all numerical values in the problem sequence X. Next, the numerical value is hidden. $h_{v_k}^n$, $h_{v_k}^c$Hidden state with original problem h_i^p , h_i^{kn} In combination, an enhanced numeric perception problem state with explicit numerical information is obtained h_i^n:

$$h_i^n = \begin{cases} \left[h_{v_k}^n : h_{v_k}^c\right] & x_i = v_k \\ \left[h_i^p : h_i^{kn}\right] & x_i \; is \; not \; a \; number \end{cases} \tag{6}$$

Then, in order to get the final number perception problem representation $\overline{h}_{na}$. We connect the hidden state of the problem h_i^p , knowledge perception problem status h_i^{kn} and number-aware problem status h_i^n , the formula is as follows:

$$\overline{h}_{na} = \left[h_i^p : h_i^{kn} : h_i^n\right] \tag{7}$$

In the decoder of the top-down solver, we directly use the tree decoder of the model, because the tree decoder has been widely used in the field of mathematical word problem solving in recent years, and the better model results also fully prove the reliability of the tree decoder. The decoder is a single layer LSTM that aims to link together the resulting digital perceptual problem representation $\overline{h}_{na}$ As input, in order to generate the target expression step by step from top to bottom, the target expression can be regarded as a pre-order traversal of the binary tree, the operator as the vertex node, the number, the operator as the leaf node, and the state is updated as follows:

$$s_{t+1} = LSTM([E(y_t) : c_t : r_t], s_t) \tag{8}$$

At time step t + 1, the decoder uses the last generated word embedding E(y_t), problem context status, c_t And expression context state r_t To update its previously hidden states_t, after which the decoder can choose from a given vocabulary V_g Generate a word. The final distribution is the combination of the generated probability and the copy probability. Since the decoder has been used many times before, the formula will not be shown in detail [10].

A numerical property prediction mechanism is added to the top-down solver, so that the solver can utilize the numerical information in the stem as efficiently as possible in the process of deriving and solving the mathematical problem, wherein, because three types of numerical properties can affect the accuracy of the solution, Therefore, we improve and add three modules related to numerical properties to improve the accuracy of the top-down solver as much as possible. They are the numerical type module, the numerical size comparison module and the numerical interference elimination module.

Value size comparison module. In the process of solving mathematical word problems, the model often encounters the "v_1-v_2" and "v_1/v_2". Because mutual subtraction and division do not have commutativity, the model decides to generate in the process of solving."v_1-v_2" or "v_2-v_1", "v_1/v_2" or "v_2/v_1" will directly affect the correctness of the final expression, and we will put every number in the question." v_k Compare with other numbers, calculate the comparison score between numbers according to the number perception problem representation z_{kj}.

At the same time, we optimize the comparison loss between numbers. Taking the case of mutual subtraction between numbers as an example, because the subtraction is rarely negative in the process of solving word problems, the larger the number is, the higher the comparison score is given:

$$g_{v_k} = \sigma\left(\mathrm{W_h}\,\overline{\mathrm{h}}_{\mathrm{v_k}}\right) \tag{9}$$

$$z_{kj} = \begin{cases} max\left(0, g_{v_j} - g_{v_k}\right) & if\ v_k \geq v_j \\ max\left(0, g_{v_k} - g_{v_j}\right) & if\ v_k < v_j \end{cases} \tag{10}$$

$$\mathcal{L}_{CR} = -\frac{1}{K^2}\sum_{k=1}^{K}\sum_{j=1}^{K} z_{kj} \tag{11}$$

Numeric type module. From the previous example, we can see "There are more 20 employees than computer consoles. How many newly recruited employees are there?" Change the number to a decimal, such as "There are more 0.25 employees than computer consoles. How many newly recruited employees are there?" Then the type of the entire final solution expression has changed, and this part of the expression has changed from "+ 20" to "* (1 + 0.25)". According to this, we divide all numbers into four types, integers, decimals, fractions, and percentages, and use the category label C to distinguish them. Given each number v_k Numerically aware problem representation $\overline{h}_{v_k}$, calculate the distribution of category scores $\mathrm{P}\left(C_{v_k} \mid \overline{h}_{v_k}\right)$, and then minimize the negative log-likelihood:

$$\mathrm{P}\left(C_{v_k} \mid \overline{\mathrm{h}}_{\mathrm{v_k}}\right) = \mathrm{softmax}\left(\mathrm{W_c}\,\overline{\mathrm{h}}_{\mathrm{v_k}}\right) \tag{12}$$

$$\mathcal{L}_{CA} = -\frac{1}{K}\sum_{k=1}^{K} \log\mathrm{P}\left(C_{v_k} \mid \overline{\mathbf{h}}_{\mathbf{v_k}}\right) \tag{13}$$

Numerical interference rejection module. At present, many researchers tend to pay attention to and make use of the relationship between numbers, but in the actual solution, it often happens that the numbers in some stem have nothing to do with the correct target expression, as shown in the previous sample, “In 2014, the fifth grade book corner of a school” The year and grade do not affect the solution of the problem, for example, if the total weight of 12 apples is 2400 grams Ask for the total price of these apples? There is a high correlation between? 12 and 2400, but the degree of influence on the solution will be different according to the different requirements of the mathematical problem. To solve this type of problem, we add a numerical interference rejection module to the top-down solver that predicts a scalar value for each number g'_{v_k}, with this scalar value indicating whether the corresponding number should appear in the derived expression, the supervision loss is defined as follows:

$$g'_{v_k} = \sigma\left(W_g\,\overline{\boldsymbol{h}}_{\boldsymbol{v_k}}\right) \tag{14}$$

$$\mathcal{L}_{\mathcal{GR}} = -\frac{1}{K}\sum_{k=1}^{K} a_i \log g'_{v_k} + (1 - a_i)\log\left(1 - g'_{v_k}\right) \tag{15}$$

Among them, when v_k to be applied to a ground truth mathematical expression, the label $a_{v_k} = 1$. Otherwise, the label is 0.

IV. Step by Step Anthropomorphic Thinking Solver

A new encoder-decoder framework is used to construct a step-by-step anthropomorphic thinking solver, which not only fuses the information of the problem stem into the decoding process, but also retains the order invariance of addition and multiplication between numbers, so that the step-by-step anthropomorphic thinking solver can recognize different forms of sub-expressions with the same meaning following the commutative law of mathematical addition and multiplication. In order to reduce the probability of generating wrong sub-expressions in the process of solving, the decoder follows the anthropomorphic step-by-step thinking in the process of generating the final expression, that is, the way of solving the problem is from bottom to top, from division to total, step by step to form the final expression.

A. Design of Local Detection Software

We define a mathematical problem P in a step-by-step anthropomorphic solver as a sequence S = $\{w_1, w_2, ..., w_l\}$. For a given quantity Q = $\{q_1, q_2, ..., q_n\} \subset$ S, a fixed constant C = $\{c_1, c_2, ..., c_m\}$ and a fixed binary operator O = $\{o_1, o_2, ..., o_k\}$ from which the solver aims to generate a list of T-step mathematical expressions E = $\{e_1, e_2, ..., e_T\}$, leading to the final answer, where $e_t = (e_i, e_j, o_t)$, $i, j < t$, $e_i, e_j \in$ Q, $o_t \in$ O, e_t is the expression resulting from the application of a binary operator to a number or previously generated subexpression, and the final expression of the resulting mathematical problem is e_T.

B. Structure of Step-by-Step Anthropomorphic Thinking Solver

We apply the language model to a mathematical problem text to obtain a representation of the numbers appearing in the text. Specifically, each quantity in the question text is converted to a token "quant" before it is entered into the language model. In the experiment, we use MWP-BER language model as an encoder for the step-by-step anthropomorphic solver, generating a hidden representation for each quantity h^q. In addition to generating a representation of the numbers in the text, we also used this encoder to generate a representation h^p of the text in question. $S^q \subset$ S is defined as one complete sentence separated by a period in the question text. We apply mean pooling to all embeddings of the problem label to obtain the problem representation h^q.

The purpose of the construction module is to enumerate all possible combinations of numbers and operators in each sub-expression generation process. Each combination is a candidate expression of the sub-expression in this step. The output sub-expression will be selected from them. In order to calculate the representation of each candidate, Constructors use permutation invariant functions for binary operators that follow the commutative law, such as addition and multiplication, and

nonlinear transformations (MLP models) for operators that do not follow the commutative law, such as additive and multiplicative inverses. The core model inside the structure is Deep Sets, which is used as the core because it also has the characteristic of permutation invariance, that is, we hope that the value of a function is independent of the order of the input elements, which can make the sub-expressions of each step generated by the whole model meet the commutative law of addition and multiplication in solving mathematical problems. Thereby reducing the probability that the model generates a wrong sub-expression. After obtaining the new intermediate expressions, the scorer module determines the final output subexpression for the step by calculating the scores of each intermediate expression h^e . After obtaining the new intermediate expressions, the scorer module calculates the score for each intermediate expression based on h^p and h^e, thereby determining the final output sub-expression for each step. Once the scorer module selects the subexpression that step, in order to enhance the generation of subsequent expressions it is imperative to update the existing intermediate expressions, a process commonly referred to as "rationalization" in academic discourse.

The intermediate expressions h^e and the text representation h^p serve as fundamental principles for interpreting how numerical explanations change from h to h'. Without this step, the model has a disadvantage that if the expression representation is not updated in the deductive reasoning process, then those expressions that are initially ranked higher are always more popular than those that are ranked lower in the whole process. We use the current intermediate expression h^e and problem representation h^p to rationalize the quantity representation, the rationalizer is composed of double GRU units, and the mathematical formula is as follows:

$$(h^q)' = GRU^2_{rat}\left(h^p, \left(GRU^1_{rat}(h^q, h^e)\right)\right) \qquad (16)$$

The first GRU unit takes a numeric representation as input, the middle expressionh^eIndicates as a hidden state. The second GRU unit takes as input the representation from the first GRU unit and takes the problem representation h^pas a hidden state, with the updated representation as the final output.

In the inference process, through the above steps, we iteratively select the expression with the highest score at each step until t reaches a predefined maximum step size t. For the resulting expressione_1, e_2, ..., e_TThe expression with the highest total score is selected as the output sub-expression of the step-by-step anthropomorphic thinking solver, and its corresponding sub-answer is calculated as part of the sub-expression of the next step.

V. Consistent Comparison Learning Module

The consistent comparison learning module is mainly used for combining two solvers with different reasoning perspectives, namely a top-down solver and a step-by-step anthropomorphic thinking solver, and because the solving ideas of the two solvers are quite different, the similar and associated sub-expression steps in the two-reasoning thinking need to be aligned. Consistent contrastive learning is used to constrain and align the similar sub-expressions generated in the two solving processes, and to learn more accurate and complete expression derivation together, so as to achieve the correct rate of Math Word Problem solving that is difficult to achieve by a single solver.

VI. Experiment

We compare the dual-thinking solution model with the latest partial solution model, and achieve a more ideal solution effect. At the same time, we also design an ablation experiment and analyze it to verify the positive impact of our improvement on the performance of the overall model.

A. Implementation details

In the encoder architecture, we employed the language models MWP-BERT and RoBERTa respectively. Within the top-down solver framework, the GAT network incorporated 8 attention heads. In the step-wise human-like thinking solver, all MLPs were configured with 2 hidden layers comprising 768 units each, utilizing Rectified Linear Unit (ReLU) activation functions. Optimization was conducted using the AdamW optimizer with a learning rate set to 2e-5, spanning 550 epochs of training, with a batch size of 12. Beam search was employed with a width of 4. Experimental setups were executed on Nvidia RTX 3090 24GB GPUs. The evaluations were performed on both the Math23K and MAWPS datasets. Specifically, for assessments on the Math23K dataset, a five-fold cross-validation methodology was adopted, denoted as "Math23K*" in the tabulated results.

B. Comparison to the state-of-the-art models

In the experiment, to more fairly demonstrate the performance of the dual-thinking model, we conducted a controlled experiment using two commonly used MWP datasets in the field of Math Word Problem solving. As shown in Table 1: Math23K, which contains over 20,000 Chinese mathematical questions, and MAWPS, which contains over 2,000 English mathematical word problems.

Table 1 Datasets

Datasets	Train	Valid	Test	Avg.Sent Len	Const.	Lang.
MAWPS	1589	199	199	30.3	17	EN
Math23K	21,162	1,000	1,000	26.6	2	CN

Table 2 Answer accuracy of our model and other models on the datasets

Models	Datasets		
	Math23K	Math23K*	MAWPS
C-MWP [11]	86.1%	84.7%	89.1%
DEDUCTREASONER [12]	84.8%	82.9%	**91.2%**
MWP-BERT+A&D [13]	85.6%	84.9%	88.7%
mBERT-TD w CL [14]	83.9%	83.2%	89.3%
M-tree [15]	82.5%	81.2%	82%
Generate & Rank [16]	85.4%	84.3%	85.1%

REAL [17]	81.9%	80.8%	83.8%
GTS [18]	75.6%	74.3%	82.6%
DTS (ours)	**86.7%**	**85.2%**	90.9%

Table 2 presented delineates the comparative efficacy of our model against several control models in terms of solution accuracy. Evidently, our proposed Dual-Thinking Solving Model exhibits superior performance across all three configurations, encompassing two distinct datasets. Notably, juxtaposed with the C-MWP model of 2023, our model demonstrates a noteworthy enhancement of 0.6% in solution accuracy on the Math23K dataset. Even when subjected to rigorous five-fold cross-validation, a discernible improvement of 0.5% persists. Moreover, notwithstanding the modest scale and simplicity of the MAWPS dataset, the performance of our Dual-Thinking Solving Model notably surpasses the majority of baseline models. This commendable performance underscores the efficacy of our methodological refinements in augmenting the model's overall performance.

C. Ablation Study

In order to better understand the effects of different modules, we conducted ablation experiments on the Math23K and MAWPS datasets.

1) The impact of dual solvers on the model

As the DTS solving model involves two solvers, it is essential to investigate the positive impact of the dual-thinking solving structure on the model's solving accuracy. For ease of comparison, it is necessary to first measure the solving performance of the two solvers in different environments separately. We designed variant models based on two different solvers and conducted experiments on the Math23K and MAWPS datasets (Table 3).

Table 3 Comparison of Answer accuracy with variant models

Models	Datasets	
	Math23K	MAWPS
TTD & RoBERTa	83.6%	88.4%
SBS & MWP-BERT	85.1%	90.3%
DTS (ours)	**86.7%**	**90.9%**

2) The impact of numerical property modules on DTS:

We incorporated three numerical property modules into the top-down solver: the numerical type determination module, numerical values comparison module, and numerical interference elimination module. This was done to enable the top-down solver to more fully consider the meaning of the numbers used when generating sub-expressions, thereby reducing the probability of generating incorrect expressions and improving the overall model's solving accuracy. We conducted ablation experiments on the dataset Math23K, and the results are shown in Table 4.

Table 4 The impact of numerical property modules on the models

Models	Math23K
TTD w/o VT	83.1%
TTD w/o VC	83.3%
TTD w/o VE	83.4%
TTD	**83.6%**
DTS w/o VT	86.3%
DTS w/o VC	86.5%
DTS w/o VE	86.6%
DTS	**86.7%**

D. Case Study

We demonstrate partial expression generation examples of the DTS (Dual-Thinking Structure) model, contrasted with the C-MWP model and the REAL model. As shown in the Table 5 below, in example 1, the C-MWP model generating the sub-expression "80%+1(3/(3+2))" in the wrong direction. Our DTS model correctly predicted that the hidden condition "1" should be included as a subtrahend in the expression. Given that the two vehicles traveled the entire distance while heading towards each other and covered "80%+3/(3+2)" together, and knowing that the vehicles were 196km apart at this time, the model deduced the correct solution expression "196/(80%+(3/(3+2)-1)".

In example 2, although the REAL model correctly predicted the numerical relationship in the first half of the expression, it misunderstood the meaning of "0.25" as a multiple rather than a fraction, leading to an erroneous expression in the latter half. In contrast, our DTS model, benefiting from its dual-thinking structure and the guidance of the numerical properties module, successfully deduced the correct expression.

Table 5 Problem analysis

example 1	The passenger bus and the truck depart from stations A and B respectively, moving towards each other. They meet after 5 hours. After their meeting, they continue at their original speeds. When they are 196km apart, the truck has covered 80% of the entire distance, and the ratio of the distance covered by the passenger bus to the distance yet to be covered is 3:2. Find the distance between stations A and B.
C-MWP:	196/(80%+1-(3/(3+2))
DTS:	196/(80%+(3/(3+2)-1)
example 2	A certain company purchased a batch of monitors and computer hosts. The number of monitors and computer hosts is known to be the same. Each monitor is worth $320, and each computer hosts is worth $300. The difference between the computer hosts and the monitor is $1800. There are more 0.25 employees than computer consoles. How many newly recruited employees are there?
REAL:	2000/(320-300)+0.25
DTS:	2000/(320-300)*(1+0.25)

VII. CONCLUSIONS

This paper presents a dual-thinking structured model for solving mathematical word problems called DTS. It has two kinds of Math Word Problem solving thinking, and also integrates numerical properties modules and DeepSets models. This unique structure greatly improves the accuracy of DTS in solving mathematical problems. We conducted numerous experiments to validate the performance of the dual-thinking solving model, and the results show that the DTS model achieved excellent performance on the Math23K, MAWPS, and MathQA datasets. Future work will involve considering improvements to the model's decoder.

REFERENCES

[1] Chen, T., Kornblith, S., Norouzi, M., Hinton, G. (2020) A simple framework for contrastive learning of visual representations[C]. In: The 37th International Conference on Machine Learning. Online. pp. 1597-1607. https://doi.org/10.48550/arXiv.2002.05709

[2] Li, Z., Zhang, W., Yan, C., Zhou, Q., Li, C., Liu, H., Cao, Y. (2022) Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems[C]. In: Findings of the Association for Computational Linguistics. Dublin, Ireland. pp. 2486-2496. https://doi.org/10.18653/v1/2022.findings-acl.195

[3] Li, J., Wang, L., Zhang, J., Wang, Y., Dai, B.T., Zhang, D. (2019) Modeling intra-relation in math word problems with different functional multi-head attentions[C]. In: The 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy. pp. 6162-6167. https://doi.org/10.18653/v1/P19-1619

[4] Chiang, T.R., Chen, Y.N. (2019) Semantically-aligned equation generation for solving and reasoning math word problems[C]. In: The 2019 Conference of the North American Chapter of the Association for Computational Linguistics. Minneapolis, Minnesota. pp. 2656-2668. https://doi.org/10.18653/v1/N19-1272

[5] Wang, B., Ju, J., Fan, Y., Dai, X., Huang, S., Chen, J. (2022) Structure-unified M-tree coding solver for math word problem[C]. In: The 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates. pp. 8122-8132. https://doi.org/10.18653/v1/2022.emnlp-main.556

[6] Chatterjee, O., Pandey, I., Waikar, A., Kumar, V., Ramakrishnan, R. (2022) WARM: A weakly (+semi) supervised math word problem solver[C]. In: The 29th International Conference on Computational Linguistics. Gyeongju, Republic of Korea. pp. 4753-4764. https://aclanthology.org/2022.coling-1.421

[7] Zhu, X., Wang, J., Zhang, L., Zhang, Y., Huang, Y., Gan, R., Zhang, J., Yang, Y. (2023) Solving math word problems via cooperative reasoning induced language models[C]. In: The 61st Annual Meeting of the Association for Computational Linguistics. Toronto, Canada. pp. 4471-4485. https://doi.org/10.18653/v1/2023.acl-long.245

[8] Wu, Q., Zhang, Q., Fu, J., Huang, X. (2020) A knowledge-aware sequence-to-tree network for math word problem solving[C]. In: The 2020 Conference on Empirical Methods in Natural Language Processing. Online. pp. 7137-7146. https://doi.org/10.18653/v1/2020.emnlp-main.579

[9] Qin, J., Lin, L., Liang, X., Zhang, R., Lin, L. (2020) Semantically-aligned universal tree-structured solver for math word problems[C]. In: The 2020 Conference on Empirical Methods in Natural Language Processing. Online. pp. 3780-3789. https://doi.org/10.18653/v1/2020.emnlp-main.309

[10] Liang, Z., Zhang, J., Wang, L., Qin, W., Lan, Y., Shao, J., Zhang, X. (2022) MWP-BERT: Numeracy-augmented pre-training for math word problem solving[C]. In: The 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Seattle, Washington. pp. 997-1009. https://doi.org/10.18653/v1/2022.findings-naacl.74

[11] Liang, Z., Zhang, J., Guo, K., Wu, X., Shao, J., Zhang, X. (2023) Compositional mathematical encoding for math word problems[C]. In: Findings of the Association for Computational Linguistics. Toronto, Canada. pp. 10008-10017. https://doi.org/10.18653/v1/2023.findings-acl.635

[12] Jie, Z., Li, J., Lu, W. (2022) Learning to reason deductively: Math word problem solving as complex relation extraction[C]. In: The 60th Annual Meeting of the Association for Computational Linguistics. Dublin, Ireland. pp. 5944-5955. https://doi.org/10.18653/v1/2022.acl-long.410

[13] Liang, Z., Zhang, J., Zhang, X. (2022) Analogical math word problems solving with enhanced problem-solution association[C]. In: The 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates. pp. 9454-9464. https://doi.org/10.18653/v1/2022.emnlp-main.643

[14] Li, Z., Zhang, W., Yan, C., Zhou, Q., Li, C., Liu, H., Cao, Y. (2022) Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems[C]. In: Findings of the Association for Computational Linguistics. Dublin, Ireland. pp. 2486-2496. https://doi.org/10.18653/v1/2022.findings-acl.195

[15] Wang, B., Ju, J., Fan, Y., Dai, X., Huang, S., Chen, J. (2022) Structure-unified M-tree coding solver for math word problem[C]. In: The 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates. pp. 8122-8132. https://doi.org/10.18653/v1/2022.emnlp-main.556

[16] Shen, J., Yin, Y., Li, L., Shang, L., Jiang, X., Zhang, M., Liu, Q. (2021) Generate & rank: A multi-task framework for math word problems[C]. In: Findings of the Association for Computational Linguistics. Punta Cana, Dominican Republic. pp. 2269-2279. https://doi.org/10.18653/v1/2021.findings-emnlp.195

[17] Huang, S., Wang, J., Xu, J., Cao, D., Yang, M. (2021) Recall and learn: A memory-augmented solver for math word problems[C]. In: Findings of the Association for Computational Linguistics. Punta Cana, Dominican Republic. pp. 786-796. https://doi.org/10.18653/v1/2021.findings-emnlp.68

[18] Xie, Z., Sun, S., (2019) A goal-driven tree-structured neural model for math word problems[C]. In: The 28th International Joint Conference on Artificial Intelligence. Macao, China. pp. 5299-5305. https://doi.org/10.24963/ijcai.2019/736