Towards Interpretable Math Word Problem Solving with Grounded Linguistic Logic Reasoning

Anonymous ACL submission

Abstract

Automatically math word problem (MWP) solving is a challenging artificial intelligence task since a machine should be able to not only understand a problem comprehensively on linguistics but also the grounded math logic entailed in the problem. Recently, lots of deep learning models have made great progress in MWP solving on answer accuracy, they rely on shallow heuristics to achieve high performance, lacking of grounded math logic reasoning, which makes them uninterpretable. To address this issue and push the research boundary of MWPs to interpretable MWP solving, we construct a large-scale and high-quality MWP dataset named InterMWP which consists of 11,507 MWP data and annotates interpretable algebraic knowledge formulas as the grounded linguistic logic of each solving equation and asks for a solver to output the formulas when it decides current predicted node is an inner-node (operator) during expression reasoning. We further propose a strong baseline called InterSolver to show the effectiveness of our constructed dataset and show how to harvest these logic knowledge by fusing logic knowledge with semantic representation to improve problem solving and make a step towards providing interpretability. Experimental results show that our InterSolver has strong logical formula-based interpretability while achieving high answer accuracy simultaneously.

1 Introduction

Automatically math word problem (MWP) solving is challenging, which aims to transform the short and math-related narrative into solution equation, as illustrated in Figure 1 (a). Recently, the task of MWPs solving has attracted a lot of research attention. Researchers have proposed several approaches (Wang et al., 2017; Huang et al., 2018; Xie and Sun, 2019; Wang et al., 2019; Qin et al., 2020, 2021) to solving MWPs based on deep learning model. However, these approaches mainly rely

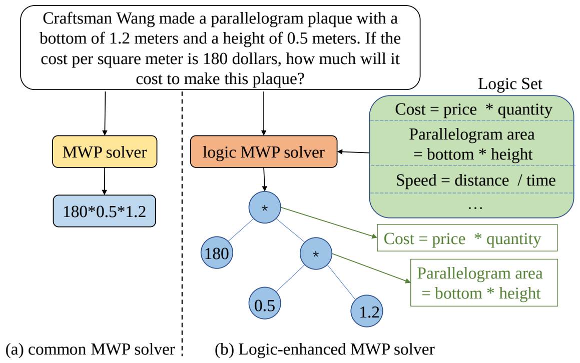


Figure 1: Common MWP dataset v.s. InterMWP dataset. Compared with the common MWP datasets, InterMWP requires a solver to predict expression tree and the corresponding linguistic logic formulas simultaneously for improving the interpretability of a solver.

on shallow heuristics to achieve high performance, lacking of grounded math logic reasoning, which makes them uninterpretable, as shown in (Patel et al., 2021). All these models can only generate solution equations directly, but they do not perceive the grounded linguistic logic implied in the problem text. For example, as shown in Figure1 (b), the grounded logic in the problem are two algebraic knowledge formulas: cost = quantity * price and parallelogram area = bottom * height where quantity is equal to parallelogram area in this MWP. Without logic reasoning, a solver is hard to explain why it generates such an equation for a solution.

To overcome this dilemma and make a step toward interpretable MWP solving, we construct a large-scale and high-quality MWP dataset called InterMWP consisting of 11,507 data samples and 210 different algebraic knowledge formulas. In InterMWP, each solution equation is annotated with interpretable algebraic knowledge formulas in tree structure as the grounded logic of each solving equation. As shown in Figure 1, each inner node is annotated with a interpretable algebraic knowledge formula which represents the grounded logic for the subtree with the current node as root ancestor.

With these logic annotations, InterMWP requires a solver to not only output the solution equation but also output the knowledge formulas simultaneously when it decides the current predicted node is an inner-node (operator) during expression reasoning. Therefore, an MWP solver developed on InterMWP can output a solution equation while generating a reasonable formula-based interpretation.

In this research, we further present a strong baseline called InterSolver to show the effectiveness of our constructed dataset and show how to harvest these logic knowledge by fusing logic knowledge with semantic representation to improve problem solving and make a step toward providing interpretability. Experimental results on InterMWP shows that our InterSolver has strong logical formula-based interpretability which achieves high answer accuracy simultaneously.

Our contributions are three-fold:

- We introduce large-scale and high-quality MWP dataset called InterMWP which makes a step towards interpretable MWP solving.
- We develop a strong baseline called InterSolver to show the effectiveness of our constructed dataset and show how to harvest these logic knowledge.
- Experimental results on InterMWP shows that our InterSolver has strong logical formula-based interpretability which achieving high answer accuracy simultaneously.

2 Related Work

2.1 Math Word Problem Solving

In recent years, deep learning-based models (Wang et al., 2017; Huang et al., 2018; Wang et al., 2018b, 2019; Xie and Sun, 2019; Chiang and Chen, 2019; Zhang et al., 2020a,b; Qin et al., 2020, 2021) have been shown impressive performance on solving MWPs by automatically learning to directly translate a problem text into an expression without any hand-crafted feature design. All these methods follow the RNN-based encoder-decoder paradigm with some different designs. Wang et al. (2017) make the first attempt to apply a vanilla sequence to sequence (seq2seq) model to translate the language text to a solution expression. Huang et al. (2018) improved their work by introducing copy and attention mechanism. Xie and Sun (2019) proposed a tree-structure decoder to decode expression as prefix order. Furthermore, Zhang et al. (2020b) improved problem text representation by fusing quantity-related graph encoder. Hong et al. (2021a) proposed to train a solver in a weakly supervised way by constructing pseudo labels during training. Hong et al. (2021b) also proposed a situation model for algebra story problems via attributed grammar. (Qin et al., 2021) proposed multiple auxiliary tasks to improved problem text representation and the ability of predicting common-sense constants. (Wu et al., 2021) enhances math word problem-solving performance by explicitly incorporating numerical values into a sequence-to-tree network and applying a numerical properties prediction mechanism. However, all these models lack interpretability so that they can give a reasonable explanation corresponding to the generated expression. To make a step towards interpretable MWP solving, we build a novel large-scale interpretable MWP dataset and propose a linguistic logic-enhanced sequence-to-tree model for generating both expression tree and corresponding formula-based interpretation.

2.2 Interpretability of MWP Solvers

Although the prior statistical models with hand-crafted features can be thought as interpretable due to the clear alignments between inputs and outputs, recently proposed deep learning approaches present new challenges to model interpretability of MWP solvers (Huang et al., 2016). (Liang et al., 2018) used pattern-matching to increasing robustness and interpretability of math word problem-solving models. (Amini et al., 2019) propose operation-based formalisms to improve the interpretability. Different from these works, we propose to predict linguistic logic along with expression construction so that our approach can explain the grounded reason about the expression generation in general with general linguistic logic formulas.

3 InterMWP Dataset

3.1 Dataset Collection

Most existing datasets for math word problem solving mainly consist of 4 parts: problem id, problem text, solution equation, and final answer, such as Math23K (Wang et al., 2017), MaWPS (Koncel-Kedziorski et al., 2016), HMWP (Qin et al., 2020), and CM17K (Qin et al., 2021). There are no explanation about why the solution equation can solve the problem, leading to a MWP solver is hard to give out the reason for constructing the solution equation. To make a step toward interpretable MWP solving, we construct a large-scale and high-quality interpretable MWP dataset called InterMWP consisting of 11,507 data samples and 210 different algebraic knowledge formulas. Excepting from the common 4 attributes as like most existing datasets, we add extra interpretable formula-based tree-structure annotation to force a MWP solver to output solving equation and grounded logic formulas simultaneously in order to make the MWP solver have a certain interpretability.

Geometric Logics
$parallelogram\ area = bottom \times height$
$rectangular\ area = length \times width$
$square\ of\ the\ radius = radius \times radius$
$circle\ area = PI\ \ \times square\ of\ the\ radius$
$cuboid\ volume = bottom\ area \times height$
Physical Logics
$speed = distance \div time$
$distance = speed \times time$
$time = distance \div speed$
$workload = time \times work\ speed$
$concentration = solute\ weight \div solution\ weight$
Financial Logics
$expenses = price \times quantity$
$insurance\ cost = insurance\ amount \times insurance\ rate$
$sales\ income = cost + profit$
$income\ after\ taxes = income\ before\ taxes - taxes$
$taxes = tax\ payable \times tax\ rate$
Commonsense Logics
$average = total \div number\ of\ units$
$total = average \times number\ of\ units$
$number\ of\ units = total \div average$
$segment\ number = interval\ points\ excluding\ both\ ends + 1$
$segment\ number = interval\ points\ including\ both\ ends - 1$

Table 1: Example logic formulas of different skills.

To collect InterMWP, we sampled 8266 examples randomly from Math23K and crawled other 3241 examples from web bank[1] to increase diversity. In total, there are 11,507 data samples in our InterMWP dataset. With these data, we first manually summarized the grounded algebraic knowledge formulas involved in the dataset into four main categries(Common-sense, Geometry, Physical, and Finance), such as cost = quantity * price, speed = distance / time, etc. Some examples are illustrated in Table 1. We summarized these formulas with general concepts so that the number of formulas can be as few as possible while covering various MWPs as more as possible. In total, there are 210 formulas summarized in InterMWP. Then, 18 well-trained annotators with undergraduate degrees manually annotated solution equation with grounded algebraic knowledge formulas in tree-structure by assigning each operator (+,-,*,/) with corresponding formula. The annotation procedure is following: 1) We use regular expressions to extract the numbers in the text and do number mapping as like (Wang et al., 2017); 2) We build the mapping between the numbers in problem and the numbers in solution equation; 3) We search adequate logic formula from 210 algebraic knowledge formulas to annotate each operator in the expression tree. An annotated data example is illustrated in Figure **??**.

[1] https://damolx.com/

3.2 Superiority of InterMWP Dataset

The superiority of our data set is mainly reflected in the following two points:

1. **Formula variables disambiguation**: As the former Math Word Problem Datasets such as Math23K (Wang et al., 2017), Alg514 (Kushman et al., 2014) and MAWPS (Koncel-Kedziorski et al., 2016)only provide a numeric expression for each problem, the reference to the variables in the formula may be ambiguous. An data example of such formula ambiguous is the problem A shown in Figure 2, original method in (Wang et al., 2017) cannot map the two numbers '2' in the equation to different positions in the problem. We overcome this shortcoming by using manpower to mapping between numbers in problem and numbers in solution equation.

2. **Complete solution set of the test split**: The former metrics to evaluate the accuracy of a MWP solver is mainly rely on the answer accuracy, but a MWP solver may output a right answer by generating a wrong formula, as the problem A shown in Figure 2, the model happened to calculate the correct answer by generate a constant number '2'. Besides, the model output of problem B in Figure 2 cannot match the original equation although they are essentially the same. To overcome this shortcoming, we use manpower to generate as many solutions as possible for each problem in the test split.

3.3 Dataset Statistics

The InterMWP dataset consists of 11,507 problems and is divided into three parts randomly: 9507 training data, 1000 validation data, and 1000 test data. The basic statistics of our InterMWP dataset is shown in Table 2. Figure 3 illustrates

Problem A: A rope is 2 decimeters long, just enough to make 2 circles around the table, what is the perimeter of the table in decimeters?
Nums_map: {N0 : 2, N1 : 2}
Equation: x = 2 / 2
Equation(ours): x = N0 / N1
Model Output(wrong): x = N0 / 2

Problem B: Xiaozhen walks to school at a speed of 3.6km/h. She arrives at school 0.25 hours after leaving home. How far is her home from school?
Nums_map: {N0 : 3.6, N1 : 0.25}
Equation: x = 3.6 * 0.25
Equation(ours): x = [N0 * N1, N1* N0]
Model Output(right): x = N1 * N0

Figure 2: Some examples compared between former MWP benchmarks and InterMWP benchmarks.

the distribution information about word-level question length, char-level question length, and expression tree length. From Figure 3, we can observe that the lengths of most of questions are adequate, which are not too long to understand for an MWP Solver. Besides, most of expression tree contains less than 3 operators, which suggests that the questions should not very difficult to reason. However, the long tail in the distribution requires the MWP solvers to understand the complex mathematical relationships in the textual content.

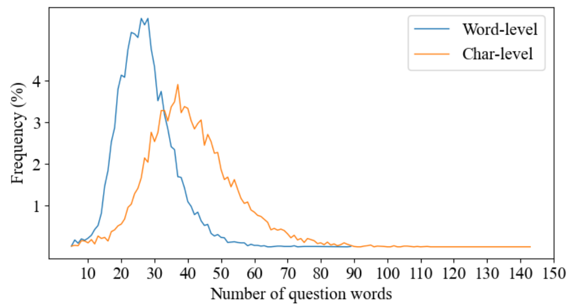
	Total	Train	Val	Test
Questions	11,507	9,507	1,000	1,000
Sentences	16,308	13,456	1,408	1,444
Words	316,620	261,700	27,048	27,872

Table 2: Basic statistics of our InterMWP dataset.

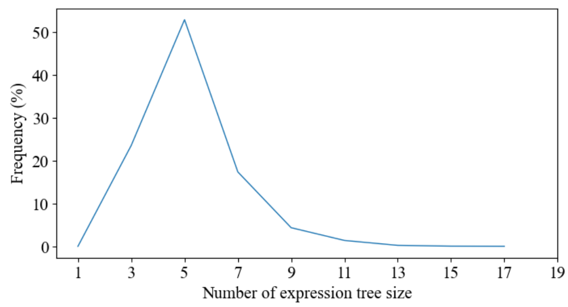
There are 210 algebraic knowledge formulas entailed in InterMWP. We list the most and least frequent knowledge formulas with a frequency greater than 5 in Table 3. It is shown that the distribution of formulas is not balanced but it is consistent with real world scene.

formulas	%
Common-sense step	56.37
average per unit = total number / number per unit	4.74
total number = average number per unit $\times$ number of units	4.74
number per unit = total number / average number per unit	2.83
...	
increased price rate = 1 + price increment ratio	0.06
increased price = original price / increased price rate	0.04

Table 3: Formulas statistics of our InterMWP dataset.



(a) Question length distribution



(b) Expression tree length distribution

Figure 3: Dataset Statistics. We show the statistical characteristics of InterMWP for intuitive observation. We can observe that out InterMWP has moderate question lenght and expression size for MWP solving.

4 InterSolver

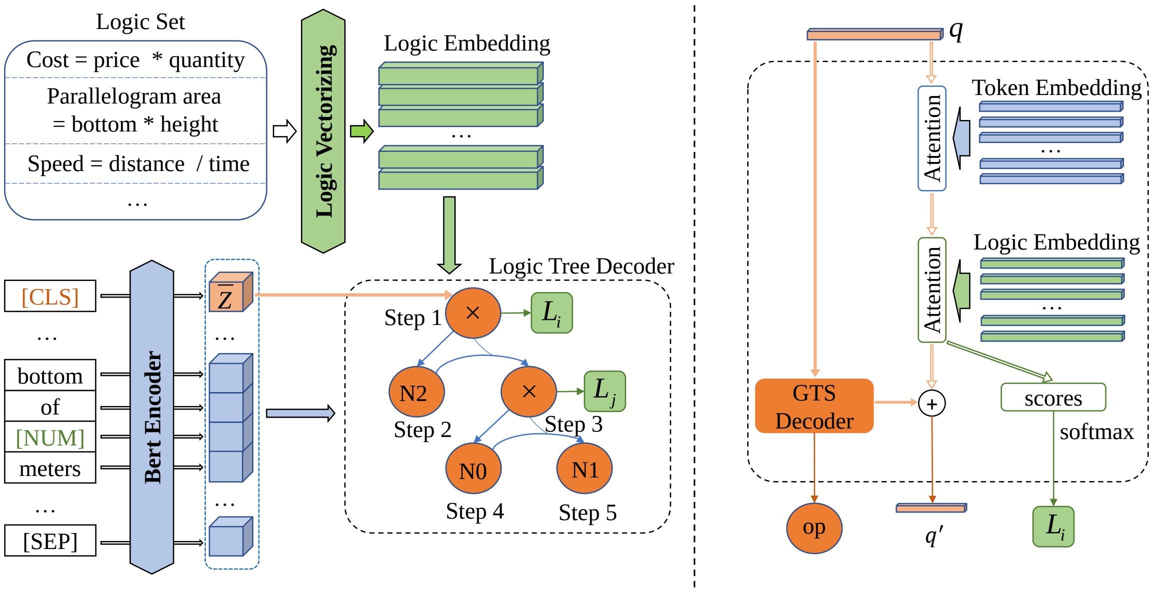
4.1 Overview

Our proposed InterSolver takes the problem text as inputs and translates them into solution expression in tree-structure and predict which formula is associated with an math operator for each inner node in the expression tree. The designed model architecture is shown in Figure 4. It contains an encoder module, a logic-enhanced tree-structure decoder module. We next introduce these modules in details.

4.2 Encoder

BERT (Devlin et al., 2019) is an efficient pre-trained language model to encode textual information, so we employ an BERTEncoder as our encoder for learning the MWP representation. The problem text sequence W is given to the BERTEncoder and transformed to the problem presentation $\bar{Z}$ and a sequence of token embeddings $\{Z_1, Z_2, ..., Z_n\}$:

$$\bar{Z}, \{Z_1, Z_2, ..., Z_n\} = BERTEncoder(W) \quad (1)$$



(a) InterSolver Framework (b) Logic Tree Decoder

Figure 4: The design of our InterSolver. (a) When a problem preprocessed by number mapping and replacement is entered, the Bert encoder encodes the problem text as context representation. Then our logic tree decoder generates an expression tree explicitly in pre-order traversal and predicts which linguistic logic can explain the current operator choice. (b) The design of our logic-enhanced tree decoder.

4.3 Logic-enhanced Tree-structure Decoder

The decoder is expected to not only output solution expression tree $Y = \{Y_1, Y_2, ..., Y_m\}$, but also predict the grounded logic formula $F = \{F_1, F_2, ..., F_k\}$ for each operator in Y. To generate solution expression tree, we following the design of the tree decoder in GTS (Xie and Sun, 2019), which predicts the pre-order traversal sequence of the expression tree. However, our decoder not only output the node of expression tree but also will predict which grounded logic formula is associated with current node if it decides current node is a math operator.

To generate the expression tree, the root node q_{root} is featured by assigning the problem representation $\bar{Z}$ of the BERTEncoder:

Then, the root node q_{root} is put in stack. The generator takes the decoding steps given next to construct the expression tree, such as, predicting the token of Y_i in the solution expression tree Y.

In each step, the prediction module first classifies the node feature q to one of token in $\{V_{num} \cup V_{op} \cup V_{con}\}$: $Y_i = nn_t(q)$, where nn_t is a two-layer neural network following (Xie and Sun, 2019). Here, V_{num}, V_{op}, and V_{con} are the set of numeric values in problem text, the set of mathematical operators, and the set of constant quantities which are occur in the solution but not in problem text. Besides, if the prediction module decides current node as an operator, it also predicts a logic formulas to explain the grounded solving logic under the current node by fusing logic formula embeddings obtained from per-trained BERT (Devlin et al., 2019), token embeddings $\{Z_1, Z_2, ..., Z_n\}$ of current problem, and current node feature q with attention mechanism (Bahdanau et al., 2015) and generate a new logic-injected node feature q':

$$
\begin{aligned}
a &= Attention(q, \{Z_1, Z_2, ..., Z_n\}) \\
\{l_1, l_2, ..., l_K\} &= Vectorizing(\{F_1, F_2, ..., F_K\}) \\
c, score &= Attention(a, \{l_1, l_2, ..., l_K\}) \\
q' &= q + c
\end{aligned} \tag{2}
$$

where $\{F_1, F_2, ..., F_K\}$ is the set of the sequences of all formulas, $\{l_1, l_2, ..., l_K\}$ is the set of logic embeddings each is generated by averaging the BERT output vectors of each token in a logic formula ($Vectorizing$), $score$ is the predicted logits for all formulas. Here, $Attention$ represents the attention mechanism implemented as following:

$$
\begin{aligned}
\mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i \\
\alpha_{t,i} &= \frac{\exp(\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_{i'}))}
\end{aligned} \tag{3}
$$

where $\boldsymbol{s}_t$ is token embedding Z_i or logic embedding l_i and $\boldsymbol{h}_i$ is node feature q or the attention result a between node feature q and token embeddings $\{Z_1, Z_2, ..., Z_n\}$.

4.3.1 Training Objective

Given the training dataset $\mathcal{D}$={(W^1, Y^1, F^1), (W^2, Y^2, F^2), $\cdots$,(W^N, Y^N, F^N) }, we minimize the following loss function:

$$\mathcal{L}(Y, F|W) = \sum_{(W,Y,F)\in\mathbf{D}} [-\log p(Y|W) + \lambda \times (-\log p(F|W))] \quad (4)$$

where

$$p(Y|W) = \prod_{t=1}^{m} \text{prob}(y_t|\mathbf{q}_t, \mathbf{c}_t, W)$$
$$p(F|W) = \prod_{t=1}^{k} \text{prob}(f_t|\mathbf{q}_t, \mathbf{c}_t, W) \quad (5)$$

where m denotes the size of Y, and $\mathbf{q}_t$ and $\mathbf{c}_t$ are the hidden state vector and its context vector at the t-th node. We set λ as 0.1 empirically.

Discussion Although our model can output expression along with corresponding formula-based interpretation for the MWP problems that has different logic formula uncovered by training set, our solver still can act as existing deep learning-based MWP solvers to generate expression with uncertain logic formulas, making our solver has ability to handle unseen problems.

5 Experiments

5.1 Experimental Setup

Datasets. We only conduct experiments on our InterMWP with train-valid-test split, since there is no existing MWP dataset with interpretation.

Baselines. We compare our InterSolver with 5 state-of-the-art models: **Math-EN** (Wang et al., 2018a): a seq2seq model with equation normalization for reducing target space. **GROUPATT** (Li et al., 2019): a math word problem solver borrowing the idea of multi-head attention from Transformer (Vaswani et al., 2017). **GTS** (Xie and Sun, 2019): a tree-structured neural network in a goal-driven manner to generate expression trees. **Graph2Tree** (Zhang et al., 2020b): an enhanced GTS with quantity graph. **GTS(BERT)**: a strong baseline we constructed by replacing RNN encoder with BERTEncoder(Devlin et al., 2019) in GTS.

Evaluation Metric. Following prior works (Wang et al., 2017; Xie and Sun, 2019; Zhang et al., 2020b), we use *answer accuracy* as the evaluation metric: if the calculated value of the predicted expression tree equals the true answer, it is thought as correct since the predicted expression is equivalent to the target expression. However, *answer accuracy* will overestimate the ability of reasonable expression generation of a MWP solver, so we also introduce *formula accuracy* to evaluate whether the generated expression is one of a set of reasonable expressions that we annotate a MWP by listing all possible solution equation manually on test set. Moreover, to measure the effectiveness of the linguistic logic, we introduce *logic accuracy*: For each data sample, if the predicted solution expression is correct and the whole predicted linguistic logic is equivalent to the target linguistic logic, we consider this sample's logic is correct.

$$\textit{logic acc} = \frac{\sum_{i=1}^{N}(\hat{Y}_i = Y_i)(\hat{F}_i = F_i)}{N} \quad (6)$$

In order to analyze the accuracy of fine-grained logic, we propose a scoring mechanism named *logic score* to evaluate the node-level logic prediction performance. On the premise that the predicted expression is correct, we evaluate each sample's node-level logic accuracy.

$$\textit{logic score} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i = Y_i)[\frac{\sum_{j=1}^{L_i}(\hat{F}_{ij} = F_{ij})}{L_i}] \quad (7)$$

where the F_{ij} represents the jth logic node for ith data sample, and the L_i represents the length of F_i.

Implementation Details. We use Pytorch[2] to implement our model on Linux with an NVIDIA RTX2080Ti GPU card. All those words with fewer than 5 occurrences are converted into a special token [UNK]. In InterSolver, the size of word embeddings and all hidden states for other layers are all set as 768, following the configuration of BERT-base (Devlin et al., 2019). In the decoder, the size of word embeddings and all hidden states for other layers are set as 128 and 768, respectively. In each epoch, all training data is shuffled randomly, and then cut into mini-batches.

InterSolver is initialized by pre-trained BERT-wwm (Cui et al., 2020) for chineses. Our InterSolver is optimized by ADAM optimizor (Kingma and Ba, 2015) with β_1 = 0.9, β_2 =0.999, and ϵ = $1e^{-8}$. The mini-batch size is set as 32. The initial

[2] http://pytorch.org

Model	value acc	formula acc	logic acc	logic score
Math-EN	62.6	58.3	-	-
Group-Attn	63.9	60.2	-	-
GTS	69.6	64.1	-	-
Graph2Tree	71.4	66.7	-	-
GTS(Bert)	80.9	75.0	-	-
InterSolver(ours)	**81.8**	**75.8**	**67.4**	**69.87**

Table 4: Various accuracies of different models on InterMWP.

fine-tuning learning rate is set as $1e^{-5}$ and $1e^{-4}$ for pretrained BERTEncoder and tree-decoder and then decreases to half every 25 epochs. To prevent overfitting, we set the dropout rate as 0.5 and weight decay as $1e^{-5}$. Finally, we use the beam search algorithm to generate expression trees and predict logic formulas.

5.2 Main Results

The main results are shown in Table 4. After injecting logic representation into our InterSolver, we increase the value accuracy of InterMWP from 80.9 to 81.8, and the formula accuracy is increased from 75.0 to 75.8, which shows that our InterSolver has stronger generalization ability than GTS(Bert). The improvement in the accuracy of traditional indicators shows that the logic formulas of our dataset helps MWPsolver to further solve the problem. Besides, our model acquired logical interpretability during training that other models do not have, our model achieve 67.4 on *logic accuracy* and 69.87 on *logic score*.

Overall, the experimental results shows the superiority of our proposed InterSolver model in both solving ability and interpretability, and the effectiveness of logic formula for solving MWPs.

5.3 Ablation on different α

We conduct experiments on different α to investigate the influences of different weights on answer accuracy. The results are shown in Table 5. From the results, we can observe that InterSolver is sensitive to different α, but it can achieve the best performance when α equals to 0.1. Therefore, we choose 0.1 as the default hyper-parameter.

α	0.00	0.05	0.10	0.15	0.20
InterSolver	80.9	81.4	**81.8**	80.2	80.6

Table 5: Ablation on different α.

Model	Commonsense	Geometric	Physical	Financial
Math-EN	56.8	54.0	86.1	58.0
Group-Attn	58.7	53.9	58.9	60.2
GTS	62.9	57.6	63.1	61.7
Graph2Tree	64.8	59.3	64.9	66.1
GTS(Bert)	73.8	**70.00**	**74.2**	73.9
InterSolver(ours)	**74.3**	68.3	**74.2**	**75.4**

Table 6: Formula accuracy on different logic skills.

Commonsense	Geometric	Physical	Financial
58.4	60.3	61.5	60.1

Table 7: InterSolver's logic accuracy on different logic skills.

5.4 Analysis on different logic classes

We also analyze the performance of our InterSolver on the four different skills(Commonsense, Geometric, Physical and Financial). As shown on Table 6. We evaluate the formula accuracy of baseline models compared with our InterSolver on the samples contained different logic class. Our InterSolver surpassed baselines model on three ascepts: Commonsense, Physical and Financial. Moreover, we also evaluated the *logic accuracy* of InterSolver on the four logic skills, the result is shown on Table 7. The result shows that InterSolver acquired good interpretability in these logic skills.

5.5 Analysis on difference logic formulas

Logics	GTS(Bert)	InterSolver
Common-sense step	75.1	75.4
total number = average number per unit × number of units	75.5	74.1
total number = number of units × average number per unit	75.3	74.3
average number per unit = total number ÷ number of units	73.5	75.9
number of unit = total number ÷ average number per unit	77.1	75.7
expenses = quantity × price	72.0	74.0
expenses = price × quantity	72.0	74.0
distance = time × speed	72.3	69.0
distance = speed × time	72.3	68.0
work speed = worklaod ÷ time	69.0	73.8

Table 8: Formula accuracy on the top-10 logic formulas with the most occurrences in the test split

We also analyze the value accuracy and formula accuracy of different logic formulas by selecting the top-10 logic formulas with the most occurrences in the dataset for comparision, the results is shown on Table 8. From the results, we can observe that InterSolver can outperform GTS(Bert) on most of top-10 logics. This shows that predicting expres-

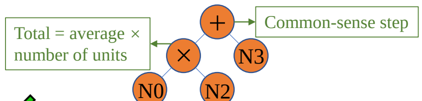
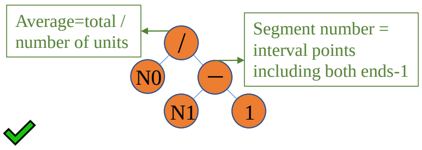
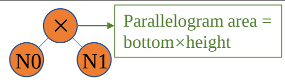
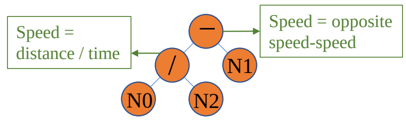
Problem	Equation	GTS(Bert)	InterSolver
A ribbon is cut every 1.4 decimetres to make 1 bow. A total of 27 bows are made. There is 1.2 decimetres left. How many decimetres is this ribbon originally?	N0 * N2 + N3, N2 * N0 + N3, N3 + N0 * N2, N3 + N2 * N0	N0 * N1 + N3 ✗	 ✓
Trees were planted on one side of a 30 meter long road. A total of 4 trees were planted from beginning to end (planting at both ends). What is the distance between two adjacent trees in meters?	N0 / (N1 - 1)	N0 / (N1 - 1) – 1 ✗	 ✓
What is the area of a parallelogram with a base of 6 cm and a height of 4 cm?	N0 * N1 N1 * N0	(N0 * N1) / N2 ✗	 ✓
The Changsha-Guangzhou railway is 728km long, and a truck runs 71km per hour from Guangzhou to Changsha. A train of passenger cars drove from Changsha to Guangzhou at the same time, and the two cars met in 4 hours. What was the speed of this train?	N0 / N2 – N1 (N0 – N1 * N2) / N2 (N0 – N2 * N1) / N2	N0 / N2 ✗	 ✓

Figure 5: Case study on GTS(Bert) and InterSolver.(Note that the results are represented as infix traversal of expression trees which is more readable than prefix traversal.)

Logics	Accuracy
Common-sense step	67.2
total number = average number per unit × number of units	56.1
total number = number of units × average number per unit	56.1
average number per unit = total number ÷ number of units	56.6
number of unit = total number ÷ average number per unit	60.0
expenses = quantity × price	56.0
expenses = price × quantity	56.0
distance = time × speed	66.0
distance = speed × time	66.0
work speed = worklaod ÷ time	45.2

Table 9: Logic accuracy on the samples contain top-10 logic formulas with the most occurrences in the test split

sion prediction and logic jointly can inject extra knowledge into MWP solver to improve problem solving. Furthermore, we also investigate the logic accuracy of the top-10 logic formulas, as shown in Table 9. We can conclude that our InterSolver can achieve acceptable accuracy, but there is still room for improvement.

5.6 Case Study

Finally, we conduct a case analysis and provide four cases in Figure 5. Benefiting from our annotated node-level logic formulas in InterMWP dataset, our InterSolver not only is more accurate on predicting operations, constants and number word, but also can extract correct logic reasoning procedure during expression tree generation. Meanwhile, GTS(Bert) is more likely to predict error expression. In summary, our InterSolver has gained a certain degree of interpretability while improving the accuracy of math word problem solving, showing the superiority of our InterMWP and InterSolver.

6 Conclusion

In this paper, we construct an interpretation math word problem dataset InterMWP which consists of 11,507 MWP data and annotates interpretable algebraic knowledge formulas as the grounded linguistic logic of each solving equation and asks for a solver to output the formulas when it decides current predicted node is an inner-node (operator) during expression reasoning. In InterMWP, we not only disambiguate the mapping between variables of formulas and the numbers in problem text, but also give a full solution equations test set to evaluate the accuracy of MWP solver's output. Besides that, we also propose a strong baseline called InterSolver which injects linguistic logic to improve the performance of MWP solving along with formula-based explanation generation. We conduct experiments on InterMWP to validate the effectiveness of our InterSolver.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *ArXiv*, abs/1905.13319.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2656–2668. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun. Zhu. 2021a. Learning by fixing: Solving math word problems with weak supervision. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, and Song-Chun. Zhu. 2021b. Smart: A situation model for algebra story problems via attributed grammar. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *international conference on learning representations*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281. Association for Computational Linguistics.

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.

Chao-Chun Liang, Yu-Shiang Wong, Yi-Chung Lin, and Keh-Yih Su. 2018. A meaning-based statistical English math word problem solver. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 652–662, New Orleans, Louisiana. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5870–5881, Online. Association for Computational Linguistics.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3780–3789, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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. 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. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5545–5552.

Lei Wang, Dongxiang Zhang, Zhang Jipeng, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Thirty-Third AAAI Conference on Artificial Intelligence*, pages 7144–7151.

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

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2021. Math word problem solving with explicit numerical values. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5859–5869, 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.