Solving Math Word Problems with Teacher Supervision

Zhenwen Liang , Xiangliang Zhang*
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
{zhenwen.liang, xiangliang.zhang}@kaust.edu.sa

Abstract

Math word problems (MWPs) have been recently addressed with Seq2Seq models by 'translating' math problems described in natural language to a mathematical expression, following a typical encoder-decoder structure. Although effective in solving classical math problems, these models fail when a delicate variation is applied to the word expression of a math problem, and leads to a remarkably different answer. We find the failure is because MWPs with different answers but similar math formula expression are encoded closely in the latent space. We thus designed a teacher module to make the MWP encoding vector match the correct solution and disaccord from the wrong solutions, which are manipulated from the correct solution. Experimental results on two benchmark MWPs datasets verified that our proposed solution outperforms the state-of-the-art models.

1 Introduction

Making computers to understand and solve math word problems (MWPs) is a very important and fundamental task of Nature Language Understanding (NLU) and has been studied a half century ago [Bobrow, 1964]. As it shows in Table 1, a math word problem usually contains text description with some given quantities, then has a question regarding an unknown quantity. The answer of the question is a mathematical formula showing how to infer the unknown quantity correctly. Building automatic MWP solvers remains a challenging task due to the wide semantic gap to parse the human-readable words into machine-understandable logics so as to facilitate quantitative reasoning [Zhang *et al.*, 2019].

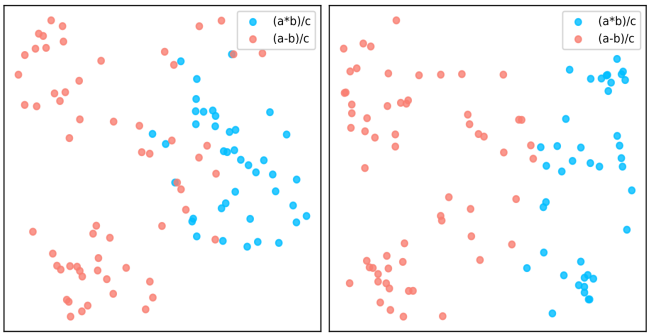
Since deep neural networks have shown their effectiveness on addressing diverse nature language processing (NLP) tasks, MWPs solvers are recently designed to 'translate' math problems into a mathematical expression via Seq2Seq models [Wang *et al.*, 2017; Wang *et al.*, 2018; Wang *et al.*, 2019; Li *et al.*, 2019; Xie and Sun, 2019; Zhang *et al.*, 2020b], which have a typical encoder-decoder structure: the *encoder* learns the representation of the problems, while the *decoder* transforms the latent representation to a math formula expression (e.g., $x = 2/10$ in Table 1). Compared with earlier semantic-parsing methods [Shi *et al.*, 2015; Koncel-Kedziorski *et al.*, 2016; Huang *et al.*, 2017] or template-based machine learning approaches [Kushman *et al.*, 2014; Hosseini *et al.*, 2014; Roy and Roth, 2018], Seq2Seq models are able to generate new formula expressions that have not been seen in the training data. Moreover, tree-structured decoder is recently introduced to model the relationship between quantities [Liu *et al.*, 2019; Xie and Sun, 2019]. To catch the relationships and order information among the quantities in MWPs, a Graph2Tree model is proposed in [Zhang *et al.*, 2020b], which has a graph-based encoder to learn the latent quantity representations and a tree-based decoder to generate solution expression trees.

Problem1:	Bob spends $2(n_0)$ hours to process $10(n_1)$ components. How many hours does he need to process one component?
Expression:	$x = (n_0/n_1)$ = 2/10
Solution:	0.2
Problem2:	Bob spends $8(n_0)$ minutes to process $1(n_1)$ component. How many component he makes per minute?
Expression:	$x = (n_1/n_0)$ = 1/8
Solution:	0.125
Graph2Tree:	$x = (n_0/n_1)$ = 8/1 (wrong)
Our Solver:	$x = (n_1/n_0)$ = 1/8 (correct)

Table 1: Two MWP examples from Math23k dataset. The solution to testing Problem2 from Graph2Tree [Zhang *et al.*, 2020b] and from our solver is presented when Problem1 is included in training.

Although the Seq2Seq based models have achieved promising results, they fail when a delicate variation is applied to the word expression of a math problem, and leads to a remarkably different answer. For example in Table 1, Problem1 and Problem2 have similar expression in text and math, but different answers. Graph2Tree [Zhang *et al.*, 2020b] trained by samples including Problem1 gives a wrong solution to Problem2. The key reason is that Seq2Seq based models do not intentionally segregate MWPs with different

*Corresponding Author



(a) (b)

Figure 1: We visualize the representations of MWPs with solution $(a * b)/c$ (blue dots) and with $(a - b)/c$ (red dots), obtained by the encoder in Graph2Tree (a) and in our proposed solver (b).

answers but similar math formula expression. Figure 1 visualizes the representation of two types of MWPs from the encoder in Graph2Tree model by T-SNE. We can see that some MWPs with solution $(a * b)/c$ have similar representation of to those with solution $(a - b)/c$. They differ only by one operator but have obviously different meanings. Therefore, we can conclude that the existing Seq2Seq based models, including Graph2Tree, concentrate only on the generation of expression, e.g., $(a * b)/c$ or $(a - b)/c$, as correct as possible from the given MWPs, and are weak on managing of distribution of MWPs representation obtained from the encoder. The overlap of two different types of MWPs can cause wrong solution generation, and lead to wrong answers.

We propose to intentionally separate the representations of MWPs with similar expression by a teacher supervision. The idea is from the process how we are taught to solve math problems. We are supervised by a human teacher to give out the correct solutions, and are also warned to avoid the wrong solutions, such that we master math problems by knowing what is correct and how the correct answer is different from the wrong ones. Therefore, we add a teacher module to make the encoder generate the representation matching the correct solution but disaccording from the wrong solutions, which are manipulated from the correct solution. By doing so, the representation of MWPs with different solutions but similar expression can be more separated, as shown in Figure 1(b).

The overall framework of our proposed solution is illustrated in Figure 2. In the training process, the encoder and decoder are jointly updated to generate correct solutions, while the encoder is additionally trained to generate high-quality representations that are examined by the teacher with both correct and manipulated wrong answers. Meanwhile, the teacher is trained to make accurate judgement about the encoded representation. Once the training is completed, the teacher module is discarded. The trained encoder-decoder takes input a given MWP and solves it by outputting the inferred solution and answer.

To evaluate the effectiveness of our proposed model, we conducted extensive experiments on two benchmark MWPs datasets and verified the following advantages of our model:

- The designed teacher module guides the encoder to learn separable representation between MWPs with similar math formula expression but markedly different answers. The resulted MWP solver outperforms state-of-the-art baselines.
- The teacher module can flexibly work with any Seq2Seq based MWP solvers, with various encoder and decoder architectures, and always improve the solvers with better encoded representation and more accurate solutions.
- The idea of presenting wrong solutions manipulated from correct solution strengthens the capability of MWP solvers. In particular, manipulating both numbers and operators in the math expression is more effective than manipulating only numbers or operators. This resonates our idea of introducing the teacher supervision to know not only what are correct and also what are wrong.

2 Related Works

MWPs were initially solved by rule-based methods [Fletcher, 1985; Bakman, 2007] and traditional statistical methods [Mitra and Baral, 2016; Kushman *et al.*, 2014]. However, those prior methods usually use a template to generate a solution, and thus require all templates to be covered in the training set. Otherwise the problems out of the training templates will be solved with wrong solutions. Recently, MWPs are addressed as a translation task, translating nature human language into mathematical language. Therefore, Seq2Seq models with encoder-decoder structure become the dominant solver. The vanilla Seq2Seq was initially employed in [Wang *et al.*, 2017], and achieved impressive performance. Then, equation normalization method [Wang *et al.*, 2018] is proposed to deal with the problem that $A + B$ and $B + A$ are both effective solutions and have the same result. Inspired by the success of Transformer [Vaswani *et al.*, 2017], multi-head attention mechanism is also applied in MWP solver [Li *et al.*, 2019]. On the one hand, the encoder in Seq2Seq is designed to learn with graphs to enrich the feature representation of MWPs in [Zhang *et al.*, 2020b]. Teacher-student network [Zhang *et al.*, 2020a] is also applied in math problem solver, which leverages knowledge distillation and differs from our approach. On the other hand, the decoder in Seq2Seq is popularly constructed as tree-based math expression generators [Wang *et al.*, 2018; Liu *et al.*, 2019]. Moreover, inspired by the human-like goal-driven study, a tree-structure recursive decoder achieves great performance and becomes a standard choice of decoder for MWPs solvers [Xie and Sun, 2019]. All above-discussed models have no control on reducing the overlap of two different types of MWPs in the problem embedding space. Our designed teacher module can be plugged into those state-of-the-art solvers and improve their performance.

3 Proposed Methods

3.1 Problem Statement

Solving MWPs is to take the input of a text sequence $W = \{w_1, w_2, ..., w_n\}$, and give the output answer sequence $A = \{A_1, A_2, ..., A_m\}$. Here n is the length of W and m is the length of A. Each w_i in the text sequence W is a word, and

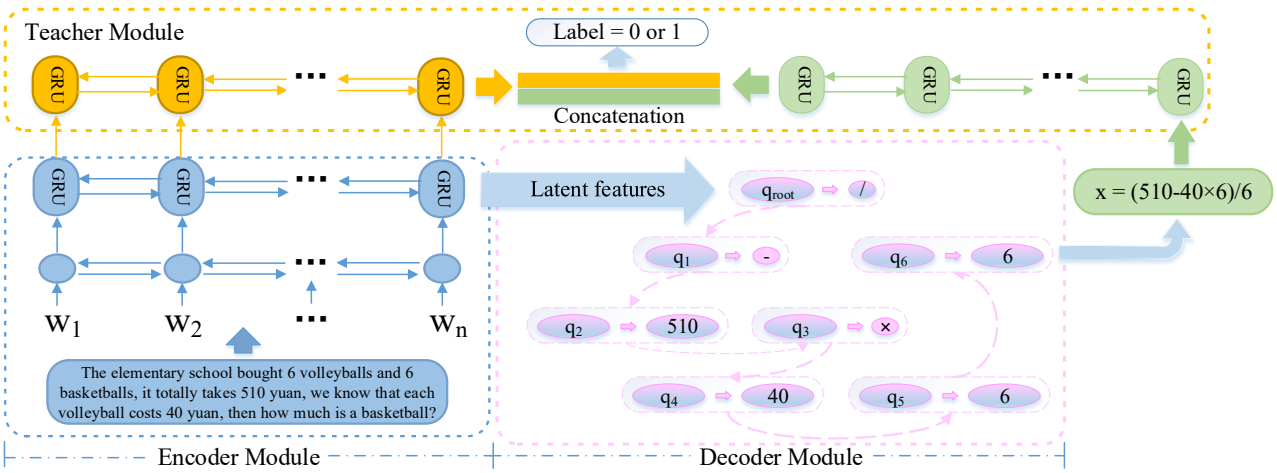


Figure 2: The proposed network architecture. The encoder module contains a bidirectional Gated Recurrent Unit (GRU) to learn latent features from the problem. The decoder module generates a tree expression of the problem. The teacher module supervises the representation learning by measuring the conformity between the learned representation and the corresponding math expression.

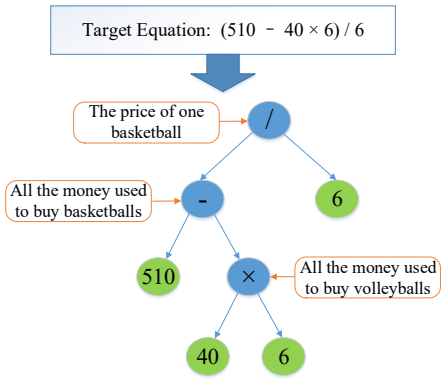


Figure 3: The description of math word problem by a tree.

A_i in the math answer sequence A is one numeric value from V_{num}, or one operator from V_{op}, or one constant from V_{con}. Note that, V_{num} contains all the numeric values that appeared in the input sequence W, defined by their orders. The set V_{op} contains all the operators in the output, e.g., '+', '-'. V_{con} contains all the constant values used in the answer, e.g., π.

3.2 Model Architecture

The designed model architecture is shown in Figure 2. It contains an *encoder* module, a *decoder* module and a *teacher* module. We next introduce these modules in details.

Encoder Module

Following the state-of-the-art MWP solver GTS model [Xie and Sun, 2019], we employ a bidirectional Gated Recurrent Unit (GRU) [Cho *et al.*, 2014] as our encoder for learning the MWP representation. Firstly, the text sequence W is presented as a sequence of the embedding vectors corresponding to the words in W. The resulted sequence W_{embed} is given to the bidirectional GRU and transformed to the representation $Z \in \mathbb{R}^{n*d}$,

$$Z = \overrightarrow{GRU}(W_{embed}) + \overleftarrow{GRU}(W_{embed}) \quad (1)$$

where d is the embedding dimension, $\overrightarrow{GRU}$ denotes the gated unit for left-to-right direction, and $\overleftarrow{GRU}$ denotes the unit for right-to-left direction.

In Graph2Tree model [Zhang *et al.*, 2020b], a graph transformer is employed as the MWP encoder for learning the representation Z. Our model is in fact flexible for the usage of any form MWP encoder, and can be deployed with the future powerful encoders, which is not the focus of this study.

Decoder Module

The decoder is expected to output answer sequence $A = \{A_1, A_2, ..., A_m\}$, from which a tree-based math expression can be built. One example of the tree is shown in Figure 3, where numeric values are at leaf nodes and operators sit at the non-leaf nodes. The tree has no parenthesis and makes the answer expressed in a simple way. The decoder in [Xie and Sun, 2019; Zhang *et al.*, 2020b] predicts the pre-order traversal sequence of the expression tree, and has been shown as an effective generator for the mathematical expression for the solution.

To generate the tree, the root node is featured by adding two last hidden states of the GRU encoder:

$$q_{root} = \overrightarrow{GRU}(W_{embed})_n + \overleftarrow{GRU}(W_{embed})_0 \quad (2)$$

where $q_{root} \in \mathbb{R}^d$ is the latent feature of the root node, $\overrightarrow{GRU}(W_{embed})_n$ is the last hidden state of forward sequence at position n, $\overleftarrow{GRU}(W_{embed})_0$ is the last state of backward sequence at position 0.

The root node (featured as q_{root}) is put in a stack. The generator takes the decoding steps given next to construct the tree, i.e., predicting the token of A_i in the answer sequence A. In each step, the prediction is in fact to classify the node feature q to one of token in $\{V_{num} \cup V_{op} \cup V_{con}\}$: $A_i = f_t(q)$. Here, f_t is implemented as a two-layer neural network following [Xie and Sun, 2019]. The numeric values in V_{num} are extracted from the W, the V_{op} is a set of known operators, and the V_{con} is given by prior knowledge.

Step 1 Pop one node from the stack. Predict the token of current node, if the prediction A_i is a number (in V_{num} or V_{con}), repeat step 1 until the stack is empty. If the prediction is an operator (in V_{op}), go to step 2.

Step 2 Generate two children nodes for the current node. For the left child, calculate its representation $q_l = f_L(q_p, Z)$, where q_p is the parent node representation and Z is the encoder output. For the right child, calculate its representation $q_r = f_R(q_p, Z, q_l)$. Here, f_L and f_R are implemented as a two-layer neural network with gating mechanism following [Xie and Sun, 2019]. Then, push the right child into stack. Predict the token of the left child node. If the prediction is a number, go to step 1. If the prediction is an operator, repeat step 2.

Teacher Module

With the designed encoder and decoder module, the typical Seq2Seq-based MWP solvers (e.g., [Xie and Sun, 2019; Zhang *et al.*, 2020b] optimize the neural network parameters by maximizing the probability of generating the sequences A with input W in training data. Our proposed teacher module additionally examines the conformity between the representation Z of input problem W and the ground-truth sequence A.

As shown in Figure 2, an answer sequence A is sent to a bidirectional GRU, for getting its representation $h_A \in \mathbb{R}^{m*d}$:

$$h_A = \overrightarrow{GRU}(A) + \overleftarrow{GRU}(A). \tag{3}$$

The representation h_A can then be compared with the representation Z for measuring their conformity, since they represent the same math problem but are obtained from two different views. Thus, the mean vector of h_A and Z are concatenated and sent to a fully-connected (FC) layer. Their conformity is predicted as a binary classification problem:

$$y = FC([\bar{Z} : \bar{h}_A]) \tag{4}$$

where $\bar{Z} \in \mathbb{R}^d$ and $\bar{h}_A \in \mathbb{R}^d$ are the mean vector of h_A and Z, respectively.

Considering to extract higher level features about the problem W, we stack another layer of bidirectional GRU on Z, and have the second option for measuring the conformity,

$$\begin{aligned} h_Z &= \overrightarrow{GRU}(Z) + \overleftarrow{GRU}(Z) \\ y &= FC([\bar{h}_Z : \bar{h}_A]) \end{aligned} \tag{5}$$

where $h_Z \in \mathbb{R}^{n*d}$ is the representation of problem W from two-layer bidirectional GRU, and $\bar{h}_Z \in \mathbb{R}^d$ is the mean vector over the n hidden states (corresponding to n words). In our evaluation results, we compared these two options and show that the higher-level representation h_Z is more comparable to h_A than Z in the conformity measurement.

To train the FC classifier, both positive examples and negative examples should be provided. Positive examples are those Z and A pairs from training data. Negative samples should be specially designed for meeting our teacher supervision target. In pairwise ranking recommendation systems, negative samples are shown to play important roles for promote the recommendation results [Rendle *et al.*, 2009]. In our case, one simple idea is to randomly select an answer from a different problem to be a negative sample of the given problem W. However, the selected answer can be markedly different from the positive answer A, making it as a too easy negative sample to be distinguished from the positive. Then, the classifier works for a too simple task as a supervisor. To this end, we design a negative answer generation algorithm for manipulating the positive answer A to get a valuable negative answer A^{neg} for the given problem, as shown in Algorithm 1. The parameter λ is a disturbance probability, deciding if changing the current token in A or not. Empirically, we set λ as 0.1. For each positive answer, we can generate different number of negative answers for training the teacher module. The evaluation results reported later show that one negative answer is enough.

Algorithm 1 Negative Answer Generation

Input: Positive answer $A = \{A_1, A_2, ..., A_m\}$
Parameter: Disturbance probability λ
Output: Negative answer $A^{neg} = \{A_1^{neg}, ..., A_m^{neg}\}$

```
Let A^neg = A
while A^neg == A do
  for i = 1 to m do
    p ← a random value between 0 and 1
    if p<λ then
      if A_i ∈ V_num then
        A_i^neg ← a random element from V_num \ A_i
      else if A_i ∈ V_op then
        A_i^neg ← a random element from V_op \ A_i
      end if
    end if
  end for
end while
return A^neg
```

3.3 Learning Objectives

Loss Function for Teacher Module. The teacher module conducts binary classification for the positive and negative samples. The objective is to minimize the binary cross-entropy loss:

$$\begin{aligned} L_{teacher}(\theta) = &- \log P(y = 1|A, Z, \theta) \\ &- \log P(y = 0|A^{neg}, Z, \theta) \end{aligned} \tag{6}$$

where θ denotes the network parameters to optimize in the teacher module, including those in the bidirectional GRU and the FC classifier network.

Loss Function for Encoder-Decoder Module. Given a training pair (W, A), the Encoder-Decoder module mainly targets on maximizing the probability of generating A for W. Moreover, with the participation of the teacher module, the encoder should maximize conformity of the representation Z and the answer A. Thus, the loss function to minimize is:

$$\begin{aligned} L_{encoder-decoder}(\phi) = &- \log P(A|W, \phi) \\ &- \alpha \times \log P(y = 1|A, Z, \theta) \end{aligned} \tag{7}$$

where ϕ denotes the network parameters in the Encoder-Decoder module, and parameter α tunes the weight of teacher module loss in the training process.

4 Experimental Settings

4.1 Dataset

We take two widely used benchmark datasets for the experimental evaluation, Math23K and MAWPS.

Math23k. Math23k [Wang *et al.*, 2017] is one of the most commonly used dataset for MWP solver evaluation. It has 23161 math word problems. Each of these problems has one ground truth mathematical expression and answer value. A test set including 1000 problems is provided in Math23k. Most of the prior work evaluates their MWP solvers on this test set. There are also others using a 5-fold cross validation to measure the performance of their solvers. In our experiments, we report the accuracy for both settings.

MAWPS. MAWPS [Koncel-Kedziorski *et al.*, 2016] is a relatively small dataset which only contains 2373 problems. We also perform 5-fold cross validation on this dataset. To make a thorough evaluation, the performance of our solvers and baseline solvers are also evaluated on this dataset. However, the ablation study and case study are conducted on the larger dataset Math23k.

4.2 Implementation Details

The embedding dimension of W_{embed} is set to 128. The latent feature dimension d is set to 512. Our models are trained for 120 epochs. The training was conducted in two stages. **Stage 1**-minimizing Eq.(6) and the first term of Eq.(7): we train the teacher and the encoder-decoder module separately at first. **Stage 2**-minimizing Eq.(6) and Eq.(7): after 30 epochs when the teacher was able to give a reasonable guidance to encoder-decoder, we apply teacher loss (i.e., the second term in Eq.(7)). We use Adam optimizer [Kingma and Ba, 2014] with initial learning rate 0.001, which is halved every 30 epochs. Dropout [Hinton *et al.*, 2012] on embedding matrix of probability 0.5 is employed to prevent overfitting. During testing, we use beam search of size 8 to generate the math expression sequence. The weight α in Eq. (7) is set to 0.1, by its sensitivity analysis. Our code in Python with Pytorch framework can be found at https://github.com/derderking/MWP-teacher.

4.3 Baselines

We conduct a comprehensive comparison with the state-of-the-art baselines, including:

- Deep Neural Solver (DNS) [Wang *et al.*, 2017], a vanilla Seq2Seq model;
- Math-EN [Wang *et al.*, 2018], employing math equation normalization to reduce the space of solution;
- T-RNN [Wang *et al.*, 2019], applying tree-structured templates;
- Group-ATT [Li *et al.*, 2019], employing multi-head attention;
- GTS [Xie and Sun, 2019], a goal-driven tree-structure solver with a powerful recursive decoder;

	Math23k	Math23k^{cv}	MAWPS
DNS	-	58.1	59.5
Math-EN	66.7	-	69.2
T-RNN	66.9	-	66.8
Group-ATT	69.5	66.9	76.1
GTS	75.6	74.3	82.6
Graph2Tree	77.4	75.5	83.7
GTS + Teacher	76.5	74.6	83.5
Graph2Tree+Teacher	**79.1**	**77.2**	**84.2**

Table 2: The accuracy (%) of MWP solvers on two datasets. Math23k^{cv} denotes the result of 5-fold cross validation on Math23k.

- Graph2Tree [Zhang *et al.*, 2020b], enriching the representation from encoder by using graph neural networks.

To avoid the implementation error that may cause unreproducible results of baseline models, we reported the results of these baselines from the papers where they are published. Since the testing set is fixed, we report our results on the same testing data when varying the encoder-decoder architectures. The GTS and Graph2Tree model have achieved better performance than all other baselines. We thus evaluate our approach with the encoder-decoder in GTS and Graph2Tree, to show the effectiveness of the proposed teacher supervision strategy.

The evaluation metric is solution accuracy, which is used as a common setting in MWP solver evaluation. We firstly translate the generated solution tree into a numerical value, then compare it with the ground truth value. The percentage of correctly addressed problems in the testing set is reported as accuracy.

5 Experimental Results

5.1 Accuracy Analysis

Table 2 shows the performance of baseline models, and the performance of GTS and Graph2Tree with our proposed teacher module. It is obvious that our teacher supervision can improve the accuracy of both GTS and Graph2Tree by a notable margin among different evaluation settings and datasets. For example, the accuracy on Math23k dataset of GTS model is improved by 0.9% with teacher loss, i.e., addressing 9 problems more correctly in the testing set. The accuracy of Graph2Tree model increases 1.7%, i.e., addressing 17 problems more correctly in the testing set. Although the solved problem set increased only by less than 20, this is already a big step, as the left un-solved problems are those specially challenging one. Note that the improvement of Graph2Tree over GTS was also just 1.8%. In the end, among 1000 problems in Math23K testing set, the model Graph2Tree with Teacher can correctly address 791 problems. This new state-of-the-art result is contributed by both the advanced encoder architecture in Graph2Tree, and the proposed teaching module on better learning of the that encoder. Later in case study, we will report the problems that Graph2Tree failed but Graph2Tree+Teacher addressed correctly, to further analyze the effectiveness of the proposed teacher supervision.

Baseline	Graph2Tree	77.4
Manipulation of A	only operators	78.0
	only numbers	78.5
	both	**79.1**
Number of negative samples to generate	1	**79.1**
	3	79.0
	5	78.4
	10	76.7

Table 3: The impact of generated negative samples, evaluated in Graph2Tree+Teacher.

GTS	75.6
Graph2Tree	77.4
GTS + Teacher (α = 0.01)	75.5
GTS + Teacher (α = 0.1)	**76.5**
GTS + Teacher (α = 1)	70.2
Graph2Tree + Teacher (α = 0.01)	78.0
Graph2Tree + Teacher (α = 0.1)	**79.1**
Graph2Tree + Teacher (α = 1)	73.1
GTS + Teacher with Z	75.8
Graph2Tree + Teacher with Z	77.9

Table 4: The influence of weight α in Eq. (7), and the usage of Z, in stead of h_Z (by default) in teacher classifier.

5.2 Ablation Study

To further evaluate the contribution of the teacher module, we conduct extensive ablation experiments. All the experiments here are conducted on the fixed testing Math23k dataset, while accuracy is still used as the evaluation metric.

Analysis of the Generated Negative Samples

The negative samples are generated in Algorithm 1 by manipulating the positive sample A. We compare the manipulation on different elements and report the results in Table 3. The manipulation on both operators and numbers is more effective than the manipulation of either one. This resonates our idea of introducing the teacher supervision to know not only what are correct and also what are wrong.

Regarding the number of negative samples to generate, the results in Table 3 show that using only one negative sample is sufficiently good. Too many negative samples discourage the representation learning of the encoder.

The Sensitivity Analysis of α

The weight α in Eq. (7) tunes the importance of the teacher loss. We report its sensitivity in Table 4. In general, keeping α as a relatively small value (e.g., α =0.1) is appropriate. A too large value of α will emphasize too much on the teacher loss and make the model lose control on the answer generation. However, a too small weight on teacher loss will have a limited impact on improving the accuracy of the model.

The Usage of Z and h_Z in Teacher Classifier

We also compare the two options of Eq.(4) (using Z) and Eq.(5) (using h_Z) in teacher module. The results in Table 4 show that using h_Z is the key to make the teacher module work. Because the higher level features h_Z is more appropriate for measuring the conformity with the generated A.

Training Sample:	Two teams A and B are repairing a $1400(n_0)$-meter road. Team A repairs $80(n_1)$ meters per day, and Team B repairs $60(n_2)$ meters per day. How many days will it take to finish repairing?
Ground Truth:	$x = n_0/(n_1 + n_2)$
Testing Sample:	Two teams A and B are repairing a $7.15(n_0)$-meter road. Team A repairs $0.65(n_1)$ meters per day, and Team B repairs $0.13(n_2)$ meters more than Team A per day. How many days will it take to finish repairing?
Ground truth:	$x = n_0/(n_1 + n_1 + n_2)$
Graph2Tree:	$x = n_0/(n_1 + n_2)$ (wrong)
Graph2Tree+Teacher:	$x = n_0/(n_1 + n_1 + n_2)$ (correct)

Table 5: Case study from Math23k. n_0 denotes the firstly appeared number in the problem description, n_1 is the second and so on.

5.3 Case Study

Table 1 has already shown a case study, comparing the performance of Graph2Tree and Graph2Tree+Teacher on solving a math problem that has a delicate variation from the known problem. Table 5 shows another case study, where the testing sample is varied slightly from a training sample on word expression. However, the Graph2Tree model gives a wrong solution, while the Graph2Tree+Teacher solves it correctly.

6 Conclusion

In this paper, we target on solving MWPs, especially those with similar text description, but markedly different solutions. We propose to intentionally separate their representations by a teacher supervision, which measures the conformity between the problem representation and the ground truth solution in math expression. The resulted representations of MWPs with different solutions but similar expression are more separated than those from the baseline models. Experimental results show that the teacher module can make Graph2Tree achieve new state-of-the-art performance. Moreover, our teacher loss is only computed in training phase, without any computational burden during testing. In our future work, we will explore how to use dynamic negative sampling which has been widely used for training recommendation systems to get better negative samples.

Acknowledgements

This work is supported by King Abdullah University of Science and Technology (KAUST), Saudi Arabia.

References

[Bakman, 2007] Yefim Bakman. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*, 2007.

[Bobrow, 1964] Daniel G. Bobrow. Natural language input for a computer problem solving system. Technical report, Massachusetts Institute of Technology, USA, 1964.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014.

[Fletcher, 1985] Charles R Fletcher. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571, 1985.

[Hinton *et al.*, 2012] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[Hosseini *et al.*, 2014] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533, 2014.

[Huang *et al.*, 2017] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. Learning fine-grained expressions to solve math word problems. In *EMNLP*, pages 805–814, 2017.

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

[Koncel-Kedziorski *et al.*, 2016] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. Mawps: A math word problem repository. In *NAACL*, pages 1152–1157, 2016.

[Kushman *et al.*, 2014] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. In *ACL*, pages 271–281, 2014.

[Li *et al.*, 2019] Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. Modeling intra-relation in math word problems with different functional multi-head attentions. In *ACL*, pages 6162–6167, 2019.

[Liu *et al.*, 2019] Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. Tree-structured decoding for solving math word problems. In *EMNLP-IJCNLP*, pages 2370–2379, 2019.

[Mitra and Baral, 2016] Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems. In *ACL*, pages 2144–2153, 2016.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[Roy and Roth, 2018] Subhro Roy and Dan Roth. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics*, 6:159–172, 2018.

[Shi *et al.*, 2015] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*, pages 1132–1142, 2015.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NIPS/NeurIPS*, 30:5998–6008, 2017.

[Wang *et al.*, 2017] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In *EMNLP*, pages 845–854, 2017.

[Wang *et al.*, 2018] Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. Translating a math word problem to a expression tree. In *EMNLP*, pages 1064–1069, 2018.

[Wang *et al.*, 2019] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. Template-based math word problem solvers with recursive neural networks. In *AAAI*, volume 33, pages 7144–7151, 2019.

[Xie and Sun, 2019] Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305, 2019.

[Zhang *et al.*, 2019] Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[Zhang *et al.*, 2020a] Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. Teacher-student networks with multiple decoders for solving math word problem. In *IJCAI*, pages 4011–4017, 2020.

[Zhang *et al.*, 2020b] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. Graph-to-tree learning for solving math word problems. In *ACL*, 2020.