Self-Competitive Learning for Solving Math Word Problem

Anonymous ACL submission

Abstract

Math word problem (MWP) aims to automatically solve mathematical questions given in texts. Most previous MWP models tend to fit the sole ground-truth solution provided by the dataset, without considering the diverse but equivalent solution expressions. To mitigate this issue, we propose a **s**elf-**c**ompetitive **l**earning framework (called SCL), which attempts to get different predictions and improve the generalization ability of the model by cooperatively learning a source network and a pruned competitor network. The competitor network is created by pruning a source network, which perturbs the source network's structure and is conducive to generate diverse solutions. The source network and the competitor network learn collaboratively and teach each other throughout the training process. Extensive experiments on two large-scale benchmarks demonstrate that our model substantially outperforms the strong baseline methods. In particular, our method improves the best performance (accuracy) by 8.4% (78.4% $\rightarrow$ 86.8%) for Math23k and 6.2% (70.5% $\rightarrow$ 76.7%) for Ape210K.

1 Introduction

Math word problem (MWP) is challenging and draws much attention from researchers in the field of natural language processing (Bobrow, 1964). MWP aims to automatically answer mathematical questions given in a natural language, which requires the model not only understand what facts are presented in a text, but also possess the reasoning capability to answer the mathematical question. Table 1 shows an example MWP with a mathematical problem and its solution expression.

Inspired by the success of deep learning, the encoder-decoder framework with attention mechanisms (Bahdanau et al., 2014) have been dominated in MWP (Wang et al., 2019, 2018a,b), which bring the state-of-the-art to a new level. The key idea is to use an encoder to learn representations of problem text and employ a decoder to generate the corresponding solution expression. Subsequently. several studies propose sequence-to-tree models, which explore the tree structure information presented in the text and improve the generation of solution expressions (Xie and Sun, 2019; Zhang et al., 2020b; Wu et al., 2021).

Problem: In March of this year, Uncle lee deposited 1,000 RMB into the bank for a period of 2 years at an annual interest rate of 4.40%. How much does Uncle lee's interest pay at maturity?
Solution Expression: $1000\times4.40\%\times2$ **Solution:** 88

Table 1: A typical math word problem.

However, the previous MWP methods tend to fit the sole ground-truth solution expression provided by the datasets without considering the diverse but equivalent solution expressions. This may limit the generalization ability of the MWP methods.

Recently, some works utilize multi-decoders to encourage the MWP models to generate diverse expressions (Zhang et al., 2020a; Shen and Jin, 2020). In addition, knowledge distillation has been proven to be effective in MWP (Zhang et al., 2020a; Liang and Zhang, 2021), where a teacher model is designed to guide the learning process of a student model. As revealed in (Liang and Zhang, 2021), the teacher model can help the MWP model distinguish the MWP with similar problems but completely different solutions. However, the teacher network, which is trained on the same data with the student network, is likely to confuse similar solution expressions with the student network. In order to avoid generating the same expression, a regularized teacher model is necessary. On the other hand, the previous model (Zhang et al., 2020a) also encourages the MWP model to diversify the solutions by perturbing the latent variables in the encoder. However, as revealed in (Liang and Zhang, 2021), the solution expression is sensitive to the perturbed latent variables, since MWP with different answers but similar mathematical problems are

encoded closely in the latent space. Hence, more effective strategies need to be introduced to build the teacher and student networks for diversifying the solution expressions and establishing a robust MWP method.

In this paper, we propose a **s**elf-**c**ompetitive **l**earning framework (called SCL) to solve MWP tasks. Our SCL model attempts to produce diverse solution expressions and improve the generalization ability of the model by cooperatively learning a source network and a pruned competitor network. The competitor network is created by pruning the source network, which perturbs the source network's structure and is conducive to generate diverse solution expressions. The competitor and source networks learn collaboratively and teach each other throughout the training process.

We summarize our main contributions as follows: (1) We propose a novel self-competitive learning framework for MWP, which is conductive to generate diverse solution expressions and improves the generalisability of the MWP model. (2) We conduct extensive experiments on two benchmark datasets (i.e., Math23k and Ape210k). The results demonstrate that our model achieves significantly better performance than the strong baselines.

2 Methodology

A math word problem (MWP) can be denoted by a projection $F : W \mapsto Y$, where $W = \{w_1, w_2, \ldots, w_m\}$ is the problem sequence with m words and $Y = \{y_1, y_2, \ldots, y_n\}$ is the solution expression of the problem with n words. The goal of MWP is to establish a model F which generates a correct solution expression Y and calculates the correct answer for the problem W.

As illustrated in Figure 1, the proposed SCL is composed of a source network (denoted as *S*) and a competitor network (denoted as *C*). The competitor network is obtained by pruning the source network. The two networks are optimised collaboratively and teach each other throughout the training process. We use the encoder-decoder framework as the backbone of both source and competitor networks.

2.1 The Encoder-Decoder Architecture

We briefly introduce the encoder and the decoder.

Encoder We adopt the RoBERTa model (Liu et al., 2019b) as our encoder. We pass the problem sequence W into the RoBERTa model and obtain problem representation $Z \in \mathbb{R}^{m*d}$, where d is the embedding size of the encoder. In order to model the relationship between the quantities in the pre-training model, we set up a learnable quantity embedding matrix $Q_E = \{q_1, q_2, \ldots, q_n\}$, similar to the learnable position embedding in BERT (Devlin et al., 2018). Before passing the sequence W into the encoder, we first replace the each quantity in the sequence W with a token $t_i \in Q_t$.

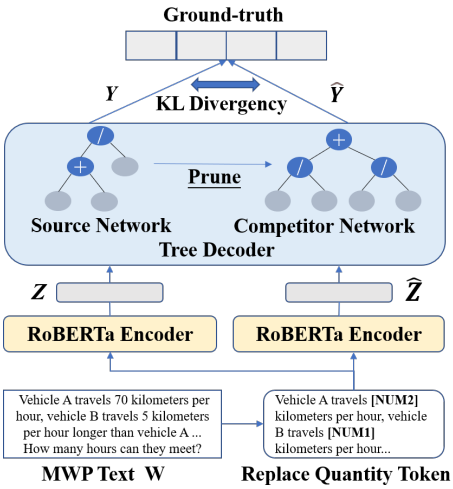

Figure 1: Overview of the proposed framework SCL

Decoder Our decoder follows the GTS model (Xie and Sun, 2019). We use the recursive operation of the decoder to construct Y by the order of pre-order traversal. First, the root node q_{root} (middle operator part) is first generated. Then, we generate the left child node q_l. Finally, we generate the right child node q_r. This process has been iterated until the leaf nodes are generated. The attention mechanism is applied to learn the global context vector G_i which is utilized to generate the current node token $\hat{Y}_i$. Here we denote the digital embedding after being encoded by the encoder as Q. Mathematically, we define the attention mechanism as follows:

$$G_i = \begin{cases} \text{Attention}\,(Z, q_{\text{root}}, q_l), & q_l \notin \emptyset. \\ \text{Attention}\,(Z, q_{\text{root}}, q_{sl}), & q_{sl} \notin \emptyset. \\ \text{Attention}\,(Z, q_{\text{root}}), & q_l, q_{sl} \in \emptyset. \end{cases} \tag{1}$$

$$\hat{Y}_i = \text{Predict}(G_i, Q). \tag{2}$$

where $\text{Predict}(\cdot)$ is the final prediction layer for producing the tree node.

If the current node is an operator, we will generate the left and right child nodes and push them into the stack in the tree decoder according to the top-down method. If it is a number, we will perform the merge operation until the leaf nodes in the stack pop out, and the result of the merge is

pushed into the left child node stack for attention operation. The merge operation will pop the required node q_{op} and q_{subtree} from an embedding stack. This recursive construction process can be defined as follows:

$$q_l = \text{Left}(\hat{G}, \hat{Y}, q_{\text{root}}). \quad (3)$$

$$q_r = \text{Right}(\hat{G}, \hat{Y}, q_{\text{root}}). \quad (4)$$

$$q_m = \text{Merge}(q_{\text{op}}, q_{\text{subtree}}, q_{m-1}). \quad (5)$$

2.2 Self-competitive Learning

As shown in Figure 1, the proposed SCL is composed of a source network and a competitor network. The competitor network is obtained by pruning the source network. In each iteration, the two networks provide each other with equivalent solution expression supervision signals. At the same time, these two networks are also trained by ground-truth supervision signals.

Formally, the training objective of the source network is to minimize negative log-likelihood (NLL) loss for each instance (W, Y) from training data:

$$\mathcal{L}_S(\theta_S) = -\sum_{i=1}^{|\mathcal{V}^o|} \mathbb{I}\{\boldsymbol{y}_i = V_i\} \log p(\boldsymbol{y}_i \mid W; \theta_S). \quad (6)$$

where $\mathbb{I}$ is an indicator function. θ_S denotes the parameters of the source network. $\mathcal{V}^o$ denotes the solution vocabulary.

We prune the model parameters θ_S of the source model and obtain the parameters θ_C of the competitor network. The training objective of the competitor network C can be defined as:

$$\mathcal{L}_C(\theta_C) = -\sum_{i=1}^{|\mathcal{V}^o|} \mathbb{I}\{\boldsymbol{y}_i = V_i\} \log p(\boldsymbol{y}_i \mid W; \theta_C). \quad (7)$$

The conventional supervised losses make the MWP model fit the sole ground-truth solution, without considering the diverse but equivalent solution expressions. To mitigate this issue, we train the competitor network and the source network collaboratively, which teach each other throughout the training process. First, we use the Kullback Leibler (KL) Divergence to measure the distance from the source network's prediction $\boldsymbol{p}_1$ to the competitor network's prediction $\boldsymbol{p}_2$ by:

$$D_{KL}(\boldsymbol{p}_2 \| \boldsymbol{p}_1) = \sum_{i=1}^{n} p_2(\hat{\boldsymbol{y}}_i) \log \frac{p_2(\hat{\boldsymbol{y}}_i)}{p_1(\boldsymbol{y}_i)}. \quad (8)$$

where n is the length of the solution expression. $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ denote the i-th ground-truth and generated tokens, respectively. Considering that KL divergence is asymmetric, we also calculate the divergence from p_2 to p_1:

$$D_{KL}(\boldsymbol{p}_1 \| \boldsymbol{p}_2) = \sum_{i=1}^{n} p_1(\hat{\boldsymbol{y}}_i) \log \frac{p_1(\hat{\boldsymbol{y}}_i)}{p_2(\boldsymbol{y}_i)}. \quad (9)$$

By averaging equations 8 and 9, we get a symmetric KL divergence, denoted as $\bar{D}_{KL}$. Note that we alternately optimize S and C in each iteration, and calculate symmetric KL divergence $\bar{D}_{KL_1}$ and $\bar{D}_{KL_2}$ for S and C, respectively. We define the overall loss functions $\mathcal{L}_S$ and $\mathcal{L}_C$ for networks S and C respectively as follows:

$$\mathcal{L}_S = \mathcal{L}_1(\theta_S) + \alpha \times \bar{D}_{KL_1}. \quad (10)$$

$$\mathcal{L}_C = \mathcal{L}_2(\theta_C) + \alpha \times \bar{D}_{KL_2}. \quad (11)$$

where α is a proportional coefficient. In this way each network learns to both correctly predict the ground-truth of training instances and match the probability estimation of its peer network.

3 Experimental Setup

Datasets We conduct experiments on two benchmark MWP datasets: Math23k (Wang et al., 2017) and Ape210k (Zhao et al., 2020). Math23k contains 22162/1000 questions for training/testing, respectively. Ape210k is composed of 166,270 questions for training, 4,157 questions for validation, and 4,159 questions for testing.

Implementation Details The word embedding size of decoder is set to 1024. We adopt RoBERTa (Liu et al., 2019b) as the problem encoder. Following Roberta's setting, the hidden size of the encoder is set to 768, and we set the hidden size of the decoder to 1024. We used Adamw (Loshchilov and Hutter, 2018) as the optimizer with the learning rate as 5e-5. The mini-batch size is set to 16. We adopt a beam search with the size of 5. Dropout (dropout rate = 0.5) is employed to avoid overfitting. For Ape210K, we set the maximum sequence length of questions as 150 and that of solution expressions as 50, similar to (Wu et al., 2021). Our model takes 80 epochs on Math23k and 50 epochs on Ape210k for convergence.

Baselines We compare our model with several strong baseline methods, including NS-Solver (Qin et al., 2021), NumS2T (Wu et al., 2021), TSN-MD (Zhang et al., 2020a), MATH-EN (Wang et al., 2018a), Multi-E/D (Shen and Jin, 2020), GTS (Xie and Sun, 2019), Tree-Decoder (Liu et al., 2019a), StackDecoder (Chiang and Chen, 2018), KAS2T (Wu et al., 2021), Graph2tree (Zhang et al., 2020b), and Ape (Zhao et al., 2020).

4 Experimental Results

Main Results The evaluation metric is answer accuracy. Table 2 show the performance com-

Models	Math23k	Math23k†	Ape210k
StackDecoder	-	65.8	52.2
Tree-Decoder	69.0	-	66.5
GTS	75.6	74.3	67.7
KAS2T	76.3	-	68.7
TSN-MD	77.4	75.1	-
Graph2Tree	77.4	75.5	-
NS-Solver	-	75.6	-
Ape	-	77.5	70.2
NumS2T	78.1	-	70.5
Multi-E/D	78.4	76.9	-
Source Network	**85.5**	**84.5**	**76.3**
Competitor Network	**86.8**	**84.6**	**76.7**

Table 2: Solution accuracy of SCL and various baselines. Note that Math23K denote results on public test set and Math23K† denote 5-fold cross-validation.

Models	Answer-ac	Equation-ac
MATH-EN	66.7	60.1
GTS	75.6	64.8
TSN-MD	77.4	65.8
Source Network	**85.5**	**73.3**
Competitor Network	**86.8**	**73.5**

Table 3: Accuracy of equation generation on Math23k.

parison of our model with baseline methods on Math23K and Ape210k, respectively. Both our source network and competitor network achieve substantially better performance than the strong competitors, verifying the effectiveness of our self-competitive learning framework. We compute the accuracy of the generated solution expression. We consider a expression as correct when the predicted expression exactly matches the annotated solution. The expression prediction accuracy is reported in Table 3. We can see that the accuracy of solution expression generation is lower than the final answer prediction accuracy, showing that our model can generate some diverse solution expressions (not included in ground-truth expressions) leading to correct answers.

Ablation Study We conduct ablation test on Math23k to analyze the impact of different components in SCL. First, we remove the mutual teaching from the source and competitor networks, denoted as source w/o MT and competitor w/o MT respectively. Second, we replace the pruned competitor network with the source network, so as to evaluate the impact of pruning (denoted as source w/o pruning and competitor w/o pruning respectively). We summarize the results in Table 4. Both the pruning strategy and mutual teaching contribute greatly to the performance of SCL.

Models	Math23k
Source network w/o pruning	84.8
Competitor network w/o pruning	85.6
Source network w/o MT	84.2
Competitor network w/o MT	84.3
Source Network	**85.5**
Competitor Network	**86.8**

Table 4: Ablation study on Math23k.

α	Math23k	Pruning Rate	Math23k
0.0005	85.1	0.1	85.8
0.005	86.8	0.2	86.8
0.05	△	0.3	84.4
-	-	0.4	84.4
-	-	0.5	△

Table 5: The Sensitivity Analysis of α and Pruning Rate. △ denote divergence.

Problem Text:	A train leaves from place A at 7 o'clock and arrives at place B at 17 o'clock. The train travels 75 kilometers per hour. How many kilometers is the distance between the two places?
Ground-truth:	$(17-7)\times 75$
Source Network:	$(17-7)\times 75$
Competitor Network:	$17\times 75-7\times 75$

Table 6: Case study from Math23k.

The Sensitivity Analysis of α We analyze the sensitivity of α on Math23k. As shown in Table 5, when α is greater than or equal to 0.05, the competitor network will not converge. We obtain the best result when $\alpha = 0.005$.

The Sensitivity Analysis of Pruning Rate We analyze the sensitivity of pruning rate on the competitor network. As shown in Table 5, when the pruning rate is greater than 0.2 (the best value), the performance of the competitor network drops quickly.

Case Study As an intuitive way to show the performance of SCL, we randomly choose one problem form Math23k and show its solution expression generated by our model. As shown in Table 6, we observe that our model can produce two different but equivalent solutions.

5 Conclusion

In this paper, we proposed a self-competitive learning framework to solve MWP tasks. Our SCL model obtained diverse predictions by cooperatively learning a source network and a pruned competitor network. Extensive experiments on two benchmark MWP datasets demonstrated the effectiveness of our model.

References

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

Daniel G Bobrow. 1964. Natural language input for a computer problem solving system.

Ting-Rui Chiang and Yun-Nung Chen. 2018. Semantically-aligned equation generation for solving and reasoning math word problems. *arXiv preprint arXiv:1811.00720*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhenwen Liang and Xiangliang Zhang. 2021. Solving math word problems with teacher supervision. *IJCAI*.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019a. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2370–2379.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. *arXiv preprint arXiv:2107.01431*.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to an expression tree. *arXiv preprint arXiv:1811.05632*.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

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

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuan-Jing 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.

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

Jipeng Zhang, Ka Wei LEE, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, Qianru Sun, et al. 2020a. Teacher-student networks with multiple decoders for solving math word problem.

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. Association for Computational Linguistics.

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