# Generalizing Math Word Problem Solvers via Solution Diversification

**Zhenwen Liang[1], Jipeng Zhang[2], Lei Wang[3], Yan Wang[4], Jie Shao[5], Xiangliang Zhang[1]***

[1] University of Notre Dame
[2] Hong Kong University of Science and Technology
[3] Singapore Management University
[4] Tencent AI Lab
[5] University of Electronic Science and Technology of China
zliang6@nd.edu, jzhanggr@conect.ust.hk, lei.wang.2019@phdcs.smu.edu.sg, yanwang.branden@gmail.com,
shaojie@uestc.edu.cn, xzhang33@nd.edu

## Abstract

Current math word problem (MWP) solvers are usually Seq2Seq models trained by the (*one-problem; one-solution*) pairs, each of which is made of a *problem* description and a *solution* showing reasoning flow to get the correct answer. However, one MWP *problem* naturally has *multiple solution* equations. The training of an MWP solver with (*one-problem; one-solution*) pairs excludes other correct solutions, and thus limits the generalizability of the MWP solver. One feasible solution to this limitation is to augment multiple solutions to a given problem. However, it is difficult to collect diverse and accurate augment solutions through human efforts. In this paper, we design a new training framework for an MWP solver by introducing a *solution buffer* and a *solution discriminator*. The *buffer* includes solutions generated by an MWP solver to encourage the training data diversity. The *discriminator* controls the quality of buffered solutions to participate in training. Our framework is flexibly applicable to a wide setting of fully, semi-weakly and weakly supervised training for all Seq2Seq MWP solvers. We conduct extensive experiments on a benchmark dataset Math23k and a new dataset named Weak12k, and show that our framework improves the performance of various MWP solvers under different settings by generating correct and diverse solutions.

## Introduction

Automatic math word problem (MWP) solving has attracted the interest of researchers for a long time. Most state-of-the-art MWP solvers (Wang, Liu, and Shi 2017; Wang et al. 2019; Liu et al. 2019; Xie and Sun 2019; Zhang et al. 2020b; Shen and Jin 2020; Hong et al. 2021a; Liang et al. 2022; Hu and Jiang 2022) are Seq2Seq models, which use an encoder to get latent representations for the problem and a decoder to generate symbolic solutions. The Seq2Seq models are typically trained by the (*one-problem; one-solution*) pairs, each of which is made of a *problem* of an MWP description and a *solution* of a corresponding ground truth equation. However, MWPs typically have multiple reasonable solutions to reach the final answer. As shown in Table 1, there exists an alternative solution that gives the same answer as the given ground truth.

---

| Problem Description: | There are 40 students taking Chinese and math exams, 25 students passed the Chinese exam, 20 students passed the math exam, 10 students failed both exams. How many students pass both exams? |
|---|---|
| Ground Truth Solution: | 25+20-(40-10) = 15 |
| Alternative Solution 1: | 25+20-40+10 = 15 |
| Alternative Solution 2: | 10+25+20-40 = 15 |
| Spurious Solution: | 25-10=15 |

Table 1: An MWP example with a ground truth solution and an alternative solution that reaches the same correct answer. It also has a spurious solution, which reaches the same correct answer but makes no sense. Our target is to improve the generalizability of an MWP solver by considering the multiple solutions to one problem and distinguishing the correct solutions from the spurious solutions.

Training an MWP solver with (*one-problem; one-solution*) pairs excludes other correct solutions, and thus limits the generalizability of the MWP solver. An intuitively better idea is to incorporate multiple solutions instead of a single solution to specific MWP in training, i.e., by data augmentation. However, it is difficult to acquire diverse and accurate augment solutions. If randomly generating solutions and then fixing the wrong ones as done in (Hong et al. 2021a), spurious solutions like "25-10=15" shown in Table 1 cannot be identified. They reach the right value answer with the quantities that appeared in the MWP description but make no sense. The inclusion of them in training is harmful to the MWP solver and eventually lowers the performance.

We are thus motivated to inject solution diversity into the MWP solver training by meanwhile making quality control on the training instances. One possible way is to label problems manually with diverse solutions. However, this is too

time-consuming and costly. Another baseline is based on mathematical transformation to generate all possible variants of ground truth. The critical disadvantage of this transformation is that it generates too many solutions and makes the training target much more ambiguous, leading to poor performance. For example, we can generate 24 potential solutions for $A + B + C + D$ by only using the commutative law. Not to mention the scenario when we consider associative and distributive law if multiplications and parentheses are included. Therefore, we only encourage the potential solutions from the MWP solver itself. In fact, to improve the generalizability of MWP solvers, we should make a trade-off between the predictability and diversity of augmented solutions. Therefore, we propose a new training framework for an MWP solver by introducing a *solution buffer* and a *solution discriminator*. The *buffer* is maintained to encourage the training data diversity by including multiple solutions generated by an MWP solver for one given problem. The *discriminator* controls the quality of buffered solutions to participate in training by calculating their qualification scores. The solutions with higher qualification scores are allowed to contribute more in future training.

The whole life-cycle of our designed training framework goes as follows: i) update the encoder and decoder parameters of the MWP solver by samples in the solution buffer in a probabilistic way, i.e., higher-quality solutions are more engaged; ii) generate new solutions to the buffer by the updated solver; and iii) training a diversity-aware discriminator to evaluate the quality of MWP solutions and go to step i). In the iterative training process, a better-trained solver generates higher-quality novel solutions. More diverse solutions train a stronger encoder and decoder, which give a more accurate probabilistic estimation of the solution quality. Since the buffered solutions participate in the training of encoder and decoder with different levels of probabilistic weights, our proposed framework well controls the augmentation quality while injecting diversity in training, regardless of the type of Seq2Seq model implemented in the solver.

We conduct extensive experiments to evaluate the effectiveness of our proposed training framework on two datasets. Math23k, the most commonly used dataset for MWP solving, is used to evaluate our training method in the fully and weakly supervised setting. We also curate a novel and large dataset named Weak12k, which has only problems annotated with answer values instead of solutions. By combining Math23k and Weak12k, we train the model in semi-weakly supervised setting, and then evaluate the performance on their testing sets separately. The experimental results show that our proposed training method can generally boost the solving accuracy for different backbone solvers in various experimental settings. The solution diversity evaluation and case studies are strong proofs to show that our method can generate multiple solutions and refine the quality of the training target.

## Related Work

### Math Word Problem Solving

After the wide usage of traditional statistical algorithms (Hosseini et al. 2014; Mitra and Baral 2016) and semantic parsing

methods (Shi et al. 2015; Huang et al. 2017; Liang et al. 2018; Zou and Lu 2019) in MWP solving, deep learning methods become dominant. (Wang, Liu, and Shi 2017) first proposed to apply sequence-to-sequence (Seq2Seq) framework to solve MWP and achieved better performance compared with previous methods. Most following works focused on the generation module. (Wang et al. 2019) proposed a two-stage decoding method to decompose goals into two parts. (Liu et al. 2019; Xie and Sun 2019) proposed to use tree structure decoder. (Chiang and Chen 2018) introduced a stack-related decoder. Multiple decoder architectures (Zhang et al. 2020a; Shen and Jin 2020) were also introduced to improve generation results. On the other hand, a couple of works (Li et al. 2019; Wang et al. 2018; Lin et al. 2021) focused on improving the encoding framework. (Zhang et al. 2020b; Shen and Jin 2020; Cao et al. 2021) chose to model quantity information with a sequential combination of RNN and GNN encoder. Besides the model architectures, there are also other interesting explorations, such as knowledge distillation (Zhang et al. 2020a), situation model (Hong et al. 2021b), syntax-semantics model(Lyu and Yu 2021), ,auxiliary training tasks (Qin et al. 2021; Piekos, Michalewski, and Malinowski 2021; Liang and Zhang 2021), explicit value encoding (Wu et al. 2021) and transfer learning(Alghamdi, Liang, and Zhang 2022). Recently, pre-trained language models (Yu et al. 2021; Huang et al. 2021; Shen et al. 2021; Li et al. 2021; Liang et al. 2022; Lan et al. 2022; Liang, Zhang, and Zhang 2022) are widely applied to encode MWPs and become the strongest baselines in terms of MWP solving accuracy. in There are some other works (Ran et al. 2019; Andor et al. 2019; Chen et al. 2020) considering the weak supervision environment in numerical understanding, however, the solution diversity in MWP is unique and under-explored.

The prior work LBF (Hong et al. 2021a) presents a weakly-supervised MWP solver by considering the diversity of solutions. Our work differs completely from it on both the training framework and the MWP setting. Our proposed training framework manages the augmented multiple solutions of one MWP with their probabilistic qualification scores. The harmful influence of spurious solutions existing in LBF is alleviated, while the diversity is injected. Our training framework is also flexibly usable in different settings, i.e. full, semi-weak and weak supervision. This is also the first work that considers both the quality and diversity of solutions.

## Methodology

### Problem Formulation

We denote an MWP description as $W = \{w_1, w_2, ..., w_m\}$ with length $m$ and its equation-shaped solution as $S = \{s_1, s_2, ..., s_n\}$ with length $n$. We let $A$ be the final answer value that can be calculated from the equation $S$. Next, we define a vocabulary for solution $S$ as $V = \{V_{op}, V_{num}, V_{con}\}$, where $V_{op} = \{+, -, \times, \div, \wedge\}$ contains the operators and $V_{con} = \{1, \pi\}$ contains the constant values that could be used in the solution. The sets $V_{num}$ are created by number mapping (Wang, Liu, and Shi 2017) and have different lengths for different problems, which contain all the numerical quantities appeared in the problem description.

Algorithm 1: Weak Data Augmentation
___
**Input**: Answer $A$ and quantity set $V_{num}$ of problem $W$
**Parameter**: Constant values $V_{con}$ and operators $V_{op}$ of $W$, maximum iteration number $MAX$
**Output**: Equation $S = \{s_1, s_2, ..., s_n\}$ for $W$
  1: Let $R_1 = V_{num} \cup V_{con}, R_2 = \{\}, R_3 = V_{num} \cup V_{con}$
  2: Let $iter = 0$ denote the current iteration number
  3: **while** TRUE **do**
  4:   **for** $i \in R_1$ **do**
  5:     **for** $j \in R_3$ **do**
  6:       **for** $op \in V_{op}$ **do**
  7:         **if** $iter = MAX$ **then**
  8:           **Return** failure
  9:         **end if**
 10:         $S \leftarrow$ connect $i$ and $j$ with operator $op$
 11:         **if** $S$ leads to $A$ **then**
 12:           **Return** $S$
 13:         **else**
 14:           Insert $S$ into $R_2$
 15:         **end if**
 16:         $iter = iter + 1$
 17:       **end for**
 18:     **end for**
 19:   **end for**
 20:   $R_3 = R_2 \cup R_3, R_1 = R_2, R_2 = \{\}$
 21: **end while**
___

The objective of an MWP solver is to translate $W$ into $S$, reaching the answer $A$. An MWP solver can be trained in different settings, by having problem-equation ($W$-$S$) pairs in a fully supervised setting, and by having problem-value ($W$-$A$) pairs in a weakly supervised setting. When having a mixture of problem-equation and problem-value pairs, we investigate a semi-weakly supervised setting. In all these settings, to leverage diverse augmented solutions, we design a buffer to store and evaluate the qualification scores of these augmented solutions. The buffer is initialized by the provided equation $S$ in the fully supervised or semi-weakly supervised setting. However, in weak supervision with only problem-value ($W$-$A$) pairs, we initialize the buffer starting from $A$ by a proposed weak data augmentation method.

## Weak Data Augmentation

The weak data augmentation (WDA) method is to search potential equations based on the given value $A$ for problem $W$. With the quantities $V_{num} \cup V_{con}$ and operators $V_{op}$ in $W$, the augmentation process can be formulated as an algorithm that receives $V_{num}$ and $A$ as inputs and generates an equation $S$, with $V_{con}$ and $V_{op}$ as parameters that can be adjusted to the dataset. Although some generated solutions might be spurious, our discriminator could assign small weights to them and alleviate the side effect. In general, our Weak Data Augmentation (WDA) is an equation-orientated neuro-symbolic search algorithm. Details of the proposed WDA can be found in the Algorithm 1.

The augmentation process is based on three sets $R_1$, $R_2$ and $R_3$, where $R_1$ contains all the newly reached solutions

in the last round, $R_2$ is a temporary set to store the new solutions in the current loop, and $R_3$ has all the potential solutions founded from the beginning. Firstly, $R_2$ is initialized as empty, while $R_1$ and $R_3$ are initialized as the combination of quantity set $V_{num}$ and constant values $V_{con}$. Then we perform two for-loops to select two equations from $R_1$ and $R_3$ and connect them with an operator in $V_{op}$ to construct a new equation $S$. We iterate $i$ and $j$ on $R_1$ and $R_3$ to avoid duplicated generations. If the new equation $S$ leads to the correct answer, we return the result and terminate the algorithm. Otherwise, $S$ will be inserted into $R_2$ for future usage (line 20 and 21). To avoid the endless trial without a successful $S$, we define a maximum iteration number $MAX$ to limit the time consumption on finding $S$, which is set as 50000 empirically. A failure is returned when no valid equation for a given problem can be found within $MAX$ iterations.

To reduce the chance of generating spurious solutions and also reduce the searching time, we also formulate several hand-crafted rules. Firstly, we avoid the equations of $a \div a$ and $a - a$ ($a$ represents a random number). Because 1 is given as a constant number and 0 is meaningless for MWPs. Secondly, we discard all solutions with only numbers in $V_{con}$, without any quantity in $V_{num}$. This is because MWPs are supposed to be solved by the quantities $V_{num}$ in the problem description, assisted by $V_{con}$ like 1 and $\pi$, not just by $V_{con}$. Thirdly, there have to be multiplication operators in the solution when constant $\pi$ appears.

## Model Architecture

Our proposed framework is shown in Figure 1. It is a general framework that can host any kind of encoder-decoder solvers.

**Encoder.**  Since GTS (Xie and Sun 2019) and MWP-BERT (Liang et al. 2022) have been recognized as the most representative MWP solvers with RNN backbone and pre-trained model backbone, respectively. They are also open-sourced and easy to reproduce the results. Therefore, they are commonly used as backbone encoders by researchers (Zhang et al. 2020b; Hong et al. 2021a), and we choose those two models as our encoders.

**Decoder.**  Tree-based solvers have been proven as an effective decoder in (Xie and Sun 2019; Zhang et al. 2020b; Liang et al. 2022). The tree structure can decompose the goal and make the answer expression simpler. Empirically, we follow the decoder implementation of (Xie and Sun 2019).

## Network Training

Besides the backbone **MWP solver** (including encoder and decoder), our training framework has a **solution buffer** and a **solution discriminator**. The **buffer** stores the training data, including those initialized from the given training data, and those generated by an optimized **MWP solver**. The generated solutions for one given problem are different from the known solution to enlarge the training data diversity. To control the quality of buffered solutions to participate in training, the **discriminator** evaluates their qualification scores. The solutions with higher qualification scores are allowed to contribute
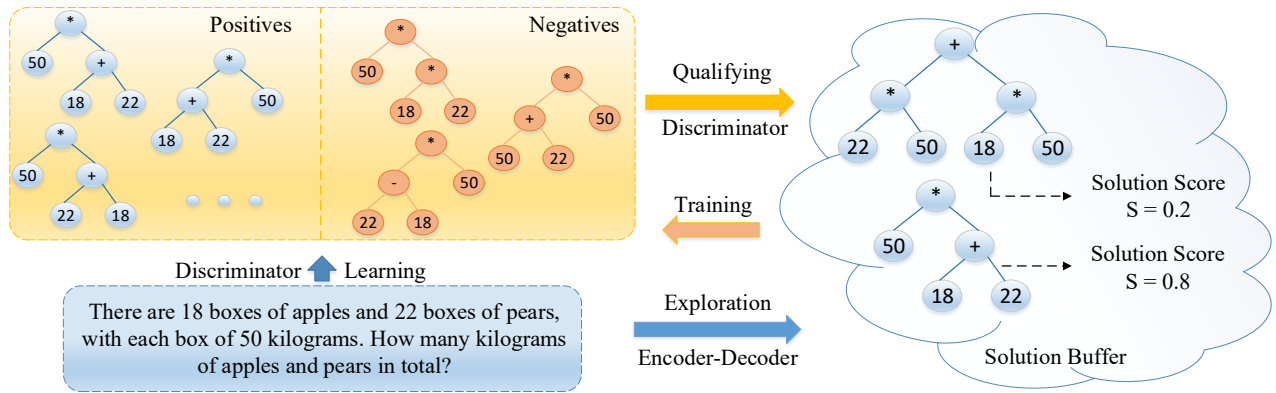
Figure 1: An overview of our proposed framework. The encoder-decoder solver is trained to generate multiple solutions to enlarge the solution buffer. Next, a discriminator is trained with positive samples and negative samples to evaluate the qualified solution. The solver is then trained again with all potential solutions in the buffer with different weights which are given by the encoder-decoder solver and the discriminator.

more in future training. The interactive process of the **MWP solver**, the **buffer** and the **discriminator** follows three steps in an iterative manner. We introduce the details of the three steps in each iteration as follows.

**Step 1: Probabilistic Training.** In a standard fully supervised setting with training problem-equation ($W$-$S$) pairs, the MWP solver parameters of both encoder and decoder can be tuned by minimizing the following negative likelihood:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\{-\log P(S|W,\theta)\}, \qquad (1)$$

where $\theta^*$ covers all the parameters of the solver and is the best to minimize the loss by finishing one entire training round at the present moment. The MWP solver with $\theta^*$ is thus supposed to recognize the correct solution $S$ for problem $W$, although its generalizability may be limited due to the training experience with only a single $S$ for each problem $W$.

The buffer in our proposed framework is designed to include diverse alternative solutions for a problem $W$. Although initialized differently in different training settings, the buffer $\mathcal{B}$ is updated during training to include more high-quality alternative solution equations. Note that at the beginning of the first Step 1 when an MWP solver with parameter $\theta$ has not been trained, the buffer $\mathcal{B}$ has only one equation of a problem $W$ or empty. In the fully supervised setting, $\mathcal{B}$ includes the given single ground-truth $S$ for $W$. In the weakly supervised setting, $\mathcal{B}$ includes the generated equation $S$ by WDA for $W$ if searchable, otherwise $\mathcal{B}$ is empty for $W$. In semi-weakly supervised setting, we only initialize the buffers of equation-annotated problems and leave others empty. Therefore, the quality score $a_i = 1$ ($a_i = 0$ when the buffer is empty) before starting to run Step 1 in the first iteration, even though at this moment $\theta$ is randomly initialized and the MWP solver performs badly on evaluating $P(B_i|W,\theta)$.

After the first iteration, the buffer is updated (step 2) and the solutions in the buffer are assigned with their qualification scores $a_i$ (step 3). With these scored solutions in buffer $\mathcal{B}$,

the MWP solver parameter $\theta$ is optimized by a new objective function:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\{-\Sigma_{B_i \in \mathcal{B}}\, a_i \log P(B_i|W,\theta)\}. \qquad (2)$$

The quality weight $a_i$ enables an alternative solution $B_i$ to participate in the training process in a probabilistic manner, instead of only focusing on the single annotated ground truth. Initially, all available $B_i$ participate with the same probability ($a_i$=1). This solution quality weight $a_i$ will be updated in Step 3.

**Step 2: Solution Buffer Update.** The trained MWP solver with optimized encoder and decoder $\theta^*$ in Step 1 is expected to generate the correct solutions for those training MWPs. To make full use of the well-trained solver, we apply $k$-beam search to generate solutions with top-$k$ probabilities. As long as the generated solutions lead to the correct value answer, we regard them as potential solutions. Then if the generated solutions do not exist in the current buffer $\mathcal{B}$, we update the buffer with the new solutions. In this way, the buffer $\mathcal{B}$ is filled by more diverse solutions.

In the initial iterations, the solver $\theta^*$ may still have limited generalizability since $\theta^*$ is optimized by using mostly (one-problem; one-solution) pairs. However, the top-$k$ generated solutions are the most likely reasonable alternative solutions since the solver $\theta^*$ does have a good understanding of the given problem. Even if some generated solutions are not good enough, our next step of qualification score evaluation can identify them and minimize their influence in the next update of $\theta^*$. After running several iterations, the MWP solver has experienced diverse solutions, improved its generalizability, and then been able to generate even more diverse solutions. This is an imitation of learning with self-correction in our human-like ways, i.e., gaining richer experience helps us to learn better.

**Step 3: Solutions Evaluation.** There are two ways to calculate the solution quality weight $a_i$ in Eq. (2). The first way is to rely on the MWP solver $\theta^*$, from which solutions are

generated. Since the generated solutions selected in buffer $\mathcal{B}$ are those with high $P(B_i|W,\theta)$, their weights are further normalized in $\mathcal{B}$ by

$$s_i = \frac{P(B_i|W,\theta)}{\Sigma_{B_j \in \mathcal{B}}P(B_j|W,\theta)}. \tag{3}$$

Staying with the calculation of $a_i = s_i$ enlarges the buffer continuously with generated solutions. The diversity is boosted, however, the quality of generated solutions will drop. To objectively evaluate the quality of one buffered solution $B_i$ for a problem $W$, we need a discriminator (classifier) to evaluate the fitness of $B_i$ and $W$, independent of the MWP solver $\theta^*$. Note that $s_i$ in Eq. (3) is proportional to the probability of mapping the problem $W$ to a solution $B_i$. It only understands (by encoding) the problem $W$ and has no understanding of $B_i$. Inspired by (Liang and Zhang 2021), we build a solution discriminator by using a contrastive learning strategy to evaluate the qualification score of $B_i$ being a correct and diverse solution to $W$.

The discriminator encodes a solution $S$ to be $Z_s$ by a bi-direction GRU, and encodes the problem $W$ to be $Z_w$ by the encoder of MWP solver. Then the score $t$ is calculated by the bilinear similarity between the mean vector of $Z_w$ and $Z_s$: $t_{ws} = \sigma(\overline{Z_w}X_t\overline{Z_s})$ where $X_t$ is a learnable matrix and $\sigma$ is the Sigmoid function. The score $t_{ws}$ should be close to 1 if $S$ is one of the correct solutions to $W$. Otherwise, $t_{ws}$ is 0. To train the discriminator for this purpose, its parameter $\phi$ (covering $X_t$ and the bi-direction GRU of solution encoder) is optimized over the following objective function:

$$\phi^* = \underset{\phi}{\mathrm{argmin}}\{- \sum_{S \in \mathcal{S}^{pos}(W)} log(t_{ws}) \\ - \sum_{S \in \mathcal{S}^{neg}(W)} log(1 - t_{ws})\} \tag{4}$$

The positive solutions $\mathcal{S}^{pos}(W)$ are generated by applying commutative law and associative law on the ground truth solution of $W$ to encourage the diversity of solutions. In weakly supervised settings, we just use the solution generated by either WDA or the model as the positive equation, because there is no ground truth solution. The negative solutions $\mathcal{S}^{neg}(W)$ are generated by following (Liang and Zhang 2021), i.e., performing random manipulations on positive solutions with a disturbance probability $\lambda$. In this way, the discriminator is trained to distinguish the diverse correct solutions and true negative solutions to problem $W$. This is the first attempt to apply operation laws to encourage the diversity of solutions in solving MWPs. We will show the effectiveness of this designed discriminator in the next section. Note that we do not perform positive sample augmentation in weakly supervised settings because there is no ground truth solution. We just use the solution generated by either WDA or the model as the positive equation. The details of the generation algorithm can be found in the next section.

**Solution Augmentation.** In order to augment the ground truth solutions to support the training of the discriminator. Let capital letters like $A, B, C$ denote a number or a complete equation that can lead to a number. For example, $A$ could be

---

**Algorithm 2: Iterative Training**

**Input**: Problem $W$, Solution Buffer $S$, Solver $M$, Discriminator $D$

1: **for** Each Problem $W$ with a solution buffer $S$ in the dataset **do**
2:    **if** Epoch $<100$ **then**
3:       Set $a_i$ in Eq. 2 as $s_i$ in Eq.3.
4:    **else**
5:       Set $a_i$ in Eq. 2 as $(s_i + t_{ws_i})/2$.
6:    **end if**
7:    Train the solver $M$ with Eq. 2 and buffer $S$.
8:    Train the discriminator $D$ with Eq. 4.
9:    Apply beam-search on $M$ to solve $W$
10:    Save equations that reach the correct value to buffer $S$
11: **end for**

---

the number 2, and also could be an equation $2+2$ that leads to 4. We design the following rule-based solution augmentation and apply them on both full solutions and partial solutions (which still have to be equations):

- For solutions with a shape $A + B$, we swap the positions $A$ and $B$ if there is no $*$ and $/$ connected to either $A$ or $B$.
- For solutions with a shape $A \pm B - C$ or $A - C \pm B$, we swap the positions of $\pm B$ and $-C$ if there is no $*$ and $/$ connected to either $B$ or $C$.
- For solutions with a shape $A * B$, we swap the positions of $A$ and $B$.
- For solutions with a shape $A*B/C$, $A/C*B$ or $A/B/C$, we swap the positions of $*B$ and $/C$ or the positions of $/B$ and $/C$.

**Training Process.** Training the discriminator by Eq. (4) needs a good MWP encoder to represent $W$ as $Z_w$. However, the problem encoder is updated with the decoder as a whole in the MWP solver by Eq. (2). In addition, the qualification score $t_{ws}$ from the discriminator is not usable to replace $a_i$ in Eq. (2) until the discriminator is well trained. Therefore, to have a stable and effective training process, we organize the training epochs in two stages. In the first stage, e.g., the first 100 epochs, the weight $a_i$ in Eq. (2) is set to $s_i$ in Eq. (3). The training process goes through step 1-3 iteratively: optimizing $\theta^*$, generating solutions to buffer, calculating $s_i$, and optimizing $\phi^*$. In the second stage when the discriminator is good enough to give reasonable scores, the weight $a_i$ is changed to $(s_i + t_{ws_i})/2$, taking into account the generation probability of the decoder and the fitness judged by the discriminator. Then the training process again goes through the iteration of optimizing $\theta^*$, generating solutions to buffer, calculating $a_i$ and optimizing $\phi^*$, until the maximum epoch is reached. For the sake of better understanding, we formulate our training process with a pseudo-code as shown in Alg. 2.

## Experiments

In this section, we first introduce the datasets and baselines that we use. Then we give a brief description about the computational environment and hyperparameters. For quantitative

| Problem Description: | Please calculate: 840/6/70+630 |
|---|---|
| Answer: | 840/6/70+630 = 632 |

Table 2: An MWP example having an explicit equation in the problem description, which are too easy, also not suitable for weak-supervised solver training and solution diversity evaluation.

analysis, our solver outperforms all baseline methods under three different supervision in terms of accuracy. Besides, we conduct a solution diversity evaluation that shows our solver is able to generate multiple solutions in beam search, which confirms that the solution diversification ability is embedded into our solver. Also, we have an ablation study to show the contribution of different components. Due to the space limit, some qualitative experiments are located in our appendix.

## Used Datasets

**D-Math23k.** Math23k contains 23,162 Chinese MWPs, which are annotated with equations as their solutions. In the Math23k dataset, there are some problems whose solutions are explicitly given in the problem description, as shown in Table 2. It is more reasonable to discard those problems without solution diversity. After cleaning, there are 22,195 MWPs left in Math23k. And we call this subset D(iversity)-Math23k. We report the performance of 5-fold cross-validation on it following (Xie and Sun 2019) and (Hong et al. 2021a). Since we filtered out many simple problems and evaluate them under a more difficult setting, i.e. 5-fold cross-validation. Therefore, our re-produced performance on D-Math23k is not as good as some baseline (Zhang et al. 2020b; Liang et al. 2022) reported.

**Weak12k.** We curate and release a novel math word problem (MWP) dataset called Weak12k with 12,117 MWPs. Each problem in this dataset is annotated with a final value answer instead of an equation solution. To our knowledge, Weak12k is the first Chinese MWP dataset in a weakly supervised manner. This dataset will be released to the public upon paper acceptance to facilitate future studies like semi-weakly supervised solver development. Compared with the most commonly used dataset Math23k, the problems in Weak12k are more difficult to solve, analysis and examples can be found in our appendix. In addition, the LBF method (Hong et al. 2021a) has a large performance gap on Math23k (over 50%) and Weak12k (below 30%) as shown in Table 4. Therefore, we believe that the new dataset Weak12k is an indispensable benchmark to the MWP community. In the experiment of this paper, we also report the results of 5-fold cross-validation on this dataset.

## Baselines

In fully supervised setting, we select DNS (Wang, Liu, and Shi 2017), S-Aligned (Chiang and Chen 2018), GTS (Xie and Sun 2019), Graph2Tree(Zhang et al. 2020b) and MWP-BERT (Liang et al. 2022) as our baselines. DNS stands for the deep neural solver, which is a vanilla GRU-based Seq2Seq

| (Fully supervised) | D-Math23k |
|---|---|
| DNS | 50.2 |
| S-Aligned | 55.4 |
| GTS | 65.7 |
| Graph2Tree | 66.6 |
| MWP-BERT | 69.2 |
| D-GTS | 67.1 |
| D-Graph2Tree | 68.9 |
| D-MWP-BERT | **73.3** |

Table 3: Comparison of answer accuracy (%) on D-Math23k dataset under the fully supervised setting. The best results are in boldface. D-solver is our proposed diversity-injected solver.

| | D-Math23k | Weak12k |
|---|---|---|
| Semi-weakly supervised | | |
| LBF | 54.1 | 33.6 |
| D-GTS w WDA | 67.1 | 57.9 |
| D-MWP-BERT w WDA | 73.2 | 67.5 |
| D-GTS w/o WDA | 68.4 | 59.2 |
| D-MWP-BERT w/o WDA | **74.4** | **70.9** |
| Weakly supervised | | |
| LBF | 53.1 | 29.5 |
| D-GTS w WDA | 55.2 | **35.0** |
| D-MWP-BERT w WDA | **56.0** | 31.8 |

Table 4: Comparison of answer accuracy (%) on D-Math23k dataset under semi-weakly and weakly supervised setting. The best results are in boldface. "w WDA" indicates that we initialize the solution buffers for problems in Weak12k with the proposed WDA. "w/o WDA" means we leave the solution buffers for Weak12k empty.

model. S-Aligned uses a stack to generate solutions. GTS presents a goal-driven tree-based solver. Graph2Tree develops a GNN-based encoder to capture more information about quantities and MWP-BERT develops a pre-trained-language-model-based encoder. For semi-weakly and weakly supervised settings, all other baselines like GTS, MWP-BERT cannot work. Therefore, we take (Hong et al. 2021a) as a baseline method, which uses a fixing mechanism to modify wrong solutions into correct ones for training.

## Implementation Details

We use Pytorch to construct the code and the NVIDIA RTX 2080Ti graphic card to train the solvers. The code and data can be found in [1]. The dimension of the embedding matrix is 128, and the dimension of all hidden features is 512. We train the model 200 epochs with the Adam optimizer (Kingma and Ba 2014) and the learning rate 0.001, which will be halved every 30 epochs. For the first 100 epochs use $a_i = s_i$ and the remaining epochs use $(s_i + t_{ws_i})/2$. We update the

---

[1]https://github.com/LZhenwen/Solution_Diversity

| | Top-1 | Top-3 | Top-5 |
|---|---|---|---|
| Fully supervised | | | |
| GTS | 65.7 | 38.3 | 29.0 |
| LBF | 65.2 | 59.3 | 51.2 |
| D-GTS | **67.1** | **61.0** | **55.9** |
| Weakly supervised | | | |
| LBF | 53.1 | 47.1 | 43.0 |
| D-GTS w WDA | **55.2** | **53.9** | **51.8** |

Table 5: Comparison of top-$k$ answer accuracy (%) on D-Math23k dataset. A higher accuracy on a larger $k$ indicates stronger generalizability to produce better and more diverse solutions.

solution buffer every 5 epochs of parameter learning, to leave sufficient time to train the model solution buffer updates.

## Quantitative Results

**Fully Supervised Training.** In this setting, we train the model with MWPs and ground truth equation solutions. Only the D-Math23k dataset is used because MWPs in Weak12k are not annotated with equations. Benefiting from our training method, the accuracies of GTS, Graph2Tree, and MWP-BERT are improved. The potential reason for such improvement is our training method enables the model to witness more diverse solutions which leads to a better generalization ability across MWPs.

**Semi-weakly Supervised Training.** In real-world scenarios, we usually get mixed data, some MWPs are annotated with equations and others are annotated with values. The baseline method LBF is not able to work well under this setting because it is specially designed for weakly supervised training only. We take a combination of D-Math23k and Weak12k for training and evaluate the answer accuracy on them separately for comparison. The results in Table 4 show that our method outperforms baseline methods by a large margin. We also find that the proposed WDA is not necessary under the semi-weakly supervised setting, since the equations generated by WDA have lower quality than those generated by trained MWP solvers in the augmentation stage. Therefore, it is better to leave the solution buffers of Weak12k MWPs empty in the beginning and fill them with solutions generated by the model trained on D-Math23k.

**Weakly Supervised Training.** Our training method is also able to work under the weakly supervised setting with the help of WDA, when no equation-annotated problems are available. As we clarified in Section 3.2, WDA is necessary for weakly supervised setting because we always need equation-annotated problems to start training. Although WDA may generate spurious solutions, the model learns from them and achieves a satisfying performance. We evaluate on D-Math23k and Weak12k datasets in this setting separately. Experimental results in Table 4 show that our proposed method is better than other baselines. The accuracy on Weak12k is lower than that on D-Math23k because the problems in Weak12k are generally more difficult to solve.
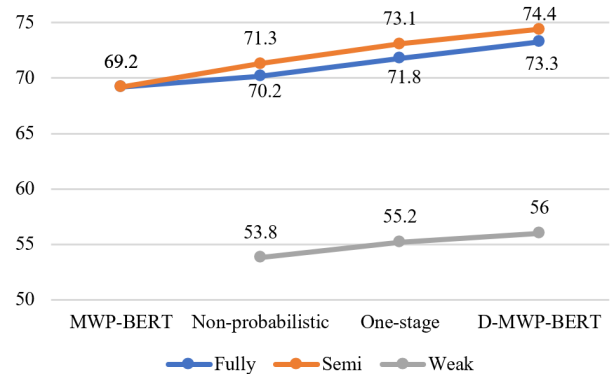


Figure 2: Ablation study under fully, semi-weakly and weakly supervised settings.

**Solution Diversity.** Following LBF (Hong et al. 2021a), we measure the overall answer accuracy in a complete beam search instead of only taking the first one. And we call this top-$k$ accuracy when the beam size is $k$. The higher accuracy of top-$k$ ($k > 1$) represents the more diverse solution that a solver can generate. The experiment results in Table 5 show that our solver outperforms GTS in the fully supervised setting and beats LBF in two different settings. We also conduct a qualitative analysis in our appendix, showing the effectiveness of our proposed solution diversification. This diversity analysis strongly demonstrates the generalizability of solution diversity that our method brings to the solver.

## Ablation Study

Some ablation on WDA is already included in Table 4. To further understand the effect of our two-stage training method, we conduct an analysis as shown in Figure 2. We use MWP-BERT as the backbone solver and conduct the experiment on the D-Math23k dataset across 3 different settings (Weak12k is also used in semi-weakly supervised setting). Non-probabilistic training means that we keep the solution buffer but assign the same weights on all solutions in the buffer, ignoring the quality differences among them. One-stage training is abandoning the qualifying stage and only using the model to score solutions. The result demonstrates that our approach is not only able to help the solver generate diverse solutions as the learning targets, but also diminish the bad effect of spurious solutions during the qualifying stage.

## Conclusion

We present a novel training framework, aiming to augment diverse solutions for MWPs and score them based on their quality. With the help of the solution buffer and the proposed buffer update method, our training framework is able to find multiple solutions for one MWP. A discriminator is trained by a contrastive learning mechanism to qualify solutions. Our method works under fully, semi-weakly, and weakly supervised situations to improve the accuracy of arbitrary MWP solvers. Moreover, we develop a simple but effective solution generation method called weak data augmentation (WDA) in the weakly supervised situation.

## Acknowledgements

## References

Alghamdi, R.; Liang, Z.; and Zhang, X. 2022. ArMATH: a Dataset for Solving Arabic Math Word Problems. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 351–362.

Andor, D.; He, L.; Lee, K.; and Pitler, E. 2019. Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension. In *EMNLP*, 5947–5952.

Cao, Y.; Hong, F.; Li, H.; and Luo, P. 2021. A bottom-up dag structure extraction model for math word problems. In *AAAI*, volume 35, 39–46.

Chen, X.; Liang, C.; Yu, A. W.; Zhou, D.; Song, D.; and Le, Q. V. 2020. Neural Symbolic Reader: Scalable Integration of Distributed and Symbolic Representations for Reading Comprehension. In *ICLR*.

Chiang, T.; and Chen, Y. 2018. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems. In *NAACL*.

Hong, Y.; Li, Q.; Ciao, D.; Huang, S.; and Zhu, S.-C. 2021a. Learning by Fixing: Solving Math Word Problems with Weak Supervision. In *AAAI*.

Hong, Y.; Li, Q.; Gong, R.; Ciao, D.; Huang, S.; and Zhu, S.-C. 2021b. SMART: A Situation Model for Algebra Story Problems via Attributed Grammar. In *AAAI*.

Hosseini, M. J.; Hajishirzi, H.; Etzioni, O.; and Kushman, N. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, 523–533.

Hu, Z.; and Jiang, M. 2022. Solving MathWord Problems Automatically with Heterogeneous Line Graph Transformer for Online Learning. *arXiv preprint arXiv:2208.05645*.

Huang, D.; Shi, S.; Lin, C.-Y.; and Yin, J. 2017. Learning fine-grained expressions to solve math word problems. In *EMNLP*, 805–814.

Huang, S.; Wang, J.; Xu, J.; Cao, D.; and Yang, M. 2021. Recall and Learn: A Memory-augmented Solver for Math Word Problems. In *Findings of EMNLP*, 786–796.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lan, Y.; Wang, L.; Zhang, Q.; Lan, Y.; Dai, B. T.; Wang, Y.; Zhang, D.; and Lim, E.-P. 2022. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. In *AAAI*, volume 36, 13188–13190.

Li, J.; Wang, L.; Zhang, J.; Wang, Y.; Dai, B. T.; and Zhang, D. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *ACL*, 6162–6167.

Li, Z.; Zhang, W.; Yan, C.; Zhou, Q.; Li, C.; Liu, H.; and Cao, Y. 2021. Seeking Patterns, Not just Memorizing Procedures: Contrastive Learning for Solving Math Word Problems. *arXiv preprint arXiv:2110.08464*.

Liang, C.-C.; Wong, Y.-S.; Lin, Y.-C.; and Su, K.-Y. 2018. A Meaning-Based Statistical English Math Word Problem Solver. In *NAACL*, 652–662.

Liang, Z.; Zhang, J.; Wang, L.; Qin, W.; Lan, Y.; Shao, J.; and Zhang, X. 2022. MWP-BERT: Numeracy-Augmented Pre-training for Math Word Problem Solving. In *Findings of NAACL*.

Liang, Z.; Zhang, J.; and Zhang, X. 2022. Analogical Math Word Problems Solving with Enhanced Problem-Solution Association. In *EMNLP*.

Liang, Z.; and Zhang, X. 2021. Solving Math Word Problems with Teacher Supervision. In *IJCAI*, 3522–3528.

Lin, X.; Huang, Z.; Zhao, H.; Chen, E.; Liu, Q.; Wang, H.; and Wang, S. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4232–4240.

Liu, Q.; Guan, W.; Li, S.; and Kawahara, D. 2019. Tree-structured decoding for solving math word problems. In *EMNLP*, 2370–2379.

Lyu, X.; and Yu, X. 2021. Solving Explicit Arithmetic Word Problems via Using Vectorized Syntax-Semantics Model. In *2021 IEEE International Conference on Engineering, Technology & Education (TALE)*, 01–07. IEEE.

Mitra, A.; and Baral, C. 2016. Learning to use formulas to solve simple arithmetic problems. In *ACL*, 2144–2153.

Piekos, P.; Michalewski, H.; and Malinowski, M. 2021. Measuring and Improving BERT's Mathematical Abilities by Predicting the Order of Reasoning. In *ACL*.

Qin, J.; Liang, X.; Hong, Y.; Tang, J.; and Lin, L. 2021. Neural-Symbolic Solver for Math Word Problems with Auxiliary Tasks. In *ACL*.

Ran, Q.; Lin, Y.; Li, P.; Zhou, J.; and Liu, Z. 2019. NumNet: Machine Reading Comprehension with Numerical Reasoning. In *EMNLP*, 2474–2484.

Shen, J.; Yin, Y.; Li, L.; Shang, L.; Jiang, X.; Zhang, M.; and Liu, Q. 2021. Generate & Rank: A Multi-task Framework for Math Word Problems. In Moens, M.; Huang, X.; Specia, L.; and Yih, S. W., eds., *EMNLP*, 2269–2279.

Shen, Y.; and Jin, C. 2020. Solving Math Word Problems with Multi-Encoders and Multi-Decoders. In *COLING*, 2924–2934.

Shi, S.; Wang, Y.; Lin, C.-Y.; Liu, X.; and Rui, Y. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*, 1132–1142.

Wang, L.; Wang, Y.; Cai, D.; Zhang, D.; and Liu, X. 2018. Translating a Math Word Problem to a Expression Tree. In *EMNLP*, 1064–1069.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B. T.; and Shen, H. T. 2019. Template-based math word problem solvers with recursive neural networks. In *AAAI*, volume 33, 7144–7151.

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep neural solver for math word problems. In *EMNLP*, 845–854.

Wu, Q.; Zhang, Q.; Wei, Z.; and Huang, X. 2021. Math Word Problem Solving with Explicit Numerical Values. In *ACL*, 5859–5869.

Xie, Z.; and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *IJCAI*, 5299–5305.

Yu, W.; Wen, Y.; Zheng, F.; and Xiao, N. 2021. Improving Math Word Problems with Pre-trained Knowledge and Hierarchical Reasoning. In *EMNLP*, 3384–3394.

Zhang, J.; Lee, R. K.-W.; Lim, E.-P.; Qin, W.; Wang, L.; Shao, J.; and Sun, Q. 2020a. Teacher-Student Networks with Multiple Decoders for Solving Math Word Problem. In *IJCAI*, 4011–4017.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020b. Graph-to-Tree Learning for Solving Math Word Problems. In *ACL*, 3928–3937.

Zou, Y.; and Lu, W. 2019. Text2Math: End-to-end Parsing Text into Math Expressions. In *EMNLP*, 5330–5340.