

Reverse Operation based Data Augmentation for Solving Math Word Problems

Qianying Liu,¹ Wenyu Guan,² Sujian Li,²
Fei Cheng,¹ Daisuke Kawahara³ and Sadao Kurohashi¹

¹ Graduate School of Informatics, Kyoto University

² Key Laboratory of Computational Linguistics, MOE, Peking University

³ School of Fundamental Science and Engineering, Waseda University

ying@nlp.ist.i.kyoto-u.ac.jp; {guanwy, lisujian}@pku.edu.cn; {feicheng, kuro}@i.kyoto-u.ac.jp; dkw@waseda.jp

Abstract

Automatically solving math word problems is a critical task in the field of natural language processing. Recent models have reached their performance bottleneck and require more high-quality data for training. Inspired by human double-checking mechanism, we propose a reverse operation based data augmentation method that makes use of mathematical logic to produce new high-quality math problems and introduce new knowledge points that can give supervision for new mathematical reasoning logic. We apply the augmented data on two SOTA math word problem solving models. Experimental results show the effectiveness of our approach¹.

Introduction

Solving Math Word Problems (MWP) is the task that infers a mathematical expression and the final answer from the natural language description of a math problem, which has been crucial due to its importance of numerically reasoning in natural language (NLP) processing (Dua et al. 2019; Nie et al. 2018). Figure 1 shows two examples which include math word problems and their corresponding solution equations and results.

The development of MWP solving has been advanced by various manually constructed MWP datasets including Math23K (Shi et al. 2015) and Dolphin18k (Shi et al. 2015). With these datasets as training data, the recent mainstream approaches are based on sequence-to-sequence (seq2seq) neural networks (NN) where a math word problem serves as the input sequence and a math equation as the output sequence. However, these approaches have reached their performance bottleneck as the size of the available data is still far from enough. Thus, how to augment data is a main consideration for improving the performance of MWP solving.

Data augmentation for NLP has been a critical and challenging research topic, especially in the case of MWP solving. Due to the preciseness of mathematics, the text description of each math word problem must be absolutely rigorous, so that even missing only one keyword could make the information incomplete and the problem unsolvable. As shown in Figure 1, all key points of the problem marked in

bold cover nearly one third of the text content. With any of the key points missed, the problem would become meaningless. Therefore, in the task of MWP solving, traditional data augmentation methods, such as randomly editing words (Wei and Zou 2019) and back translation (Yu et al. 2018), may potentially produce noise, mislead the model and degrade the performance.

In this paper, unlike previous practice of data augmentation, we instead simulate the process of human double-checking and propose an MWP generation method to obtain more high quality MWPs which are inferred through the reverse operation of the original problems. As we observe, when humans solve MWPs, a common technique to guarantee the accuracy is to perform double-checking on the problem. As shown in Figure 1, MWP1 asks about the time that the two cars have spent before meeting each other, and its solution implicitly holds the mathematical logic that $time = distance / speed$. With this logic, we can get the equation $x = 660 / (32 + 34)$, where 10 is the final answer of x and denotes the spent time. To verify the correctness of the solution of MWP1, a reverse problem MWP2 is usually conceived with the logic $distance = time * speed$, which takes the answer of MWP1 (i.e., $x = 10$) as a known quantity and the distance x' as unknown. In such a reverse operation, we can easily produce a new math word problem and its solution quality could be ensured. Concretely, we can seek one known quantity (e.g., 660 in MWP1) in the original problem and change its surrounding declarative description into an interrogative sentence (i.e., the last sentence in MWP2). At the same time, we change the original interrogative description about the unknown quantity into a declarative statement with the original solution (e.g., $x = 10$) substituted (i.e., the next to last sentence in MWP2). Then, with most content unchanged, we can obtain a new math problem (e.g., MWP2) from the original problem (e.g., MWP1).

We can see that this kind of reversion-based data augmentation has the following benefits: First, this way of generating new data is relatively simple and reliable so that the key information will not be lost; Second, the reverse operation can infer new knowledge points which give supervision for new mathematical reasoning logic; Third, more high-quality data can be used to well train the neural networks. Next, we combine this reverse operation based data augmentation (RODA) with seq2seq models and conduct ex-

MWP1	MWP2
The distance between city A and B is 660 km , the car starting from A drives 32 km/h , and the car starting from B drives 34 km/h . The two cars are starting from the two places at the same time heading toward each other . How many hours later would the two cars meet ?	The car starting from A drives 32 km/h, and the car starting from B drives 34 km/h. The two cars are starting from the two places at the same time heading toward each other. <i>10 hours later the two cars would meet. What is the distance between city A and B?</i>
Equation: $x = 660 / (32 + 34)$	Equation: $x' = 10 * (32 + 34)$
Answer: 10	Answer: 660

Figure 1: Two MWP Examples. The solution equations of MWP1 and MWP2 are reversed to each other.

periments on Math23K, which is the most influential large-scale dataset for MWPs. To be noticed, our method could be easily adapted to any supervised model on any high-quality MWP corpus. Experimental results show that our method could benefit various models and outperform previous state-of-the-art.

Related Work

Math Word Problem Solving

Early approaches of solving MWPs mainly rely on predefined rules to map the problems into several predefined templates (Bobrow 1964; Charniak 1969). Recent studies on solving MWPs adopt either semantic parsing techniques to parse the natural language questions into logic forms (Roy and Roth 2015, 2017, 2018; Shi et al. 2015; Zou and Lu 2019), or question answering style end-to-end models to directly predict the solution equations (Ling et al. 2017; Wang et al. 2019). Wang, Liu, and Shi (2017) first used seq2seq based models to directly generate the mathematical solution of MWPs. Wang et al. (2018) further extended the method by performing equation normalization as preprocessing and generating the suffix notation of the equation. Zhang et al. (2020a) further studied the diverse solution problem via multiple-decoders. Various studies (Liu et al. 2019; Wang et al. 2019; Guan et al. 2019; Chiang and Chen 2018; Xie and Sun 2019) further improved the model with tree structured information. Attention mechanism and Graph Neural Networks have been studied to capture the intra-relation of the numbers (Li et al. 2019; Zhang et al. 2020b).

Data Augmentation

The two most popular methods for sentence level data augmentation in NLP is back translation (Yu et al. 2018) and EDA (Wei and Zou 2019). Yu et al. (2018) used a high quality machine translation system to translate the original text into a new language and then backward to perform paraphrase style data augmentation. Wei and Zou (2019) slightly modified the input sentence on the token level by performing four kinds of operations: synonym replacement, random insertion, random swap, and random deletion. The drawback of applying these two methods on MWPs is that MWPs are very sensitive to even small modifications, with any piece of key information missed the problem would become unsolvable. In addition, these methods only provide lexical level variance but no new reasoning knowledge.

Recently due to the reliability of rule-based data augmentation in NLP, task-specific logic rules for data augmentation has been explored in Natural Language Inference (NLI) and Question Answering. Kang et al. (2018) studied NLI-specific logic based data augmentation which generates new examples by replacing tokens or changing labels on the original training examples. They only used the logic operations of NOT(\neg) and equivalence(\Leftrightarrow). Asai and Hajishirzi (2020) further extended entailment(\rightarrow) operation for common domain question answering data augmentation by measuring the transitive consistency of pairs of questions. Gokhale et al. (2020) studied disjunction(\vee) and conjunction(\wedge) operation for yes-no style visual question answering. All these studies involve only one or two steps simple logic. By contrast, our method makes use of reversed operation of complex mathematical computation and can introduce new reasoning logic in the generated new examples.

Methodology

Given a high-quality MWP dataset, we propose RODA which can produce new accurate math word problems to enlarge the data scale. Next we can use the augmented dataset to improve the supervised MWP solving models.

Reversion based Data Augmentation

For data augmentation, we reverse the original problems in the dataset to new problems, and the reversion process consists of three steps: number identification, problem transformation, and equation generation. To ensure the quality of the new data, the main criteria of our reversion process is “quality first quantity second”, so that our method relies on some well-designed empirical rules in the three steps.

Number Identification A MWP might contain various numbers that are irrelevant to the solution, such as the date or description text such as ‘tenth grade student’. Following Wang, Liu, and Shi (2017)’s work, we build an LSTM-based classifier to determine the significance of the numbers and filter out the irrelevant numbers².

In addition, the numbers in a math word problem do not necessarily map one to one with the numbers in the corresponding solution equation. That is, one number may appear in the solution equation, but do not appear in the word

²The classification performance can reach around 99% accuracy.

problem, and vice versa. Thus, to conduct high-quality word problem reversion, we first need to identify the valid numbers which can be converted to an unknown quantity using the reverse logic of the original solution. Here, we propose a rule-based method to identify the possible numbers for reversion. Four key rules are explained as follows, and we also show their examples in Table 1.

1. **Problem Number Duplication** If a number appears more than once in the problem, we filter this number out because we cannot map the numbers in the text to the equation with an injective function. As shown in Table 1, we cannot know whether a separate ‘2’ is related to A or B , and thus it is difficult for us to conduct a precise reversion.
2. **Equation Number Duplication** If a number appears more than once in the equation, we filter this number out because it is not capable to be solved with a linear equation with one unknown variable. Introducing new operators such as root operation would be out-of-domain (OOD) with the original data and introduce noise.
3. **Power Operation** If a number is involved with power operation, we filter it out because the inverse operation, which is logarithmic operation or root operation, is OOD. We filter out both the base number and the exponent number.
4. **Constant Term Numbers** Constant term numbers are not applicable for reverse operation, such as π and unit conversion terms.

Problem Transformation After collecting a set of valid number candidates, we perform problem reversion for each number to get a new transformed math word problem. The main work is to convert the original question sentence into a declarative sentence with a definite quantity, and convert the sentence with the identified number into a question sentence with its question point on the number. Specifically, for the first conversion, we name the original question sentence as Q . From Q we find the interrogative pronoun according to a list which is compiled in advance and replace the pronoun with the answer of the original problem. Next we adjust the word order to make the sentence fluent and natural, and get the declarative sentence D' .

For the second conversion, given the candidate number c_i , we get the sentence D which contains c_i . Next we take c_i as the question focus and change D into a question sentence Q' . For different languages, the conversion process is different. For example, for Chinese, the conversion is relatively simple and just replaces c_i with an interrogative pronoun such as “多少(How many)” without the need of adjusting word order.

From the original math word problem, we delete the two sentences D and Q , and add D' and Q' at the end of the text. Then we get the new transformed math word problem. Our problem transformation method is simple but effective, which is the basis of producing new high-quality MWP.

Equation Generation For each new transformed word problem, we need to generate its solution equation. Similar to problem transformation, the new solution equation is de-

rived according to the solution equation of the original problem.

To clarify the process of generating the new equation, we take the two MWPs in Figure 1 as examples. For the pre-identified number 660 in MWP1, it will be changed into a variable (i.e., x'). At the same time, we substitute the original variable x with its answer 10. Then we can get an intermediate equation $10 = x'/(32 + 34)$.

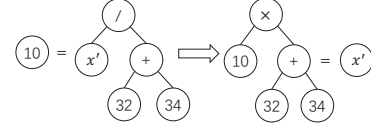


Figure 2: Example of equation conversion.

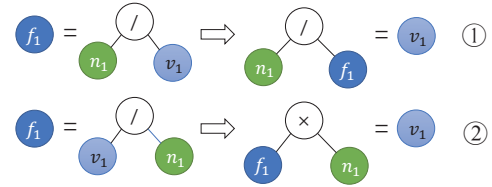


Figure 3: Two example rules of equation reversion

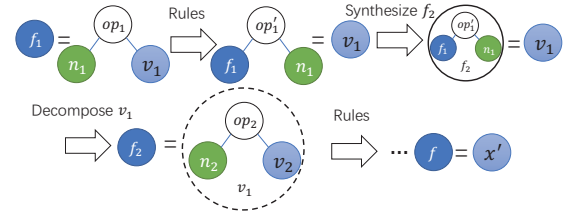


Figure 4: Illustration of recursive equation conversion.

Next we need to convert this equation to its equivalent format where the variable x' is located on the right side of the equal-sign alone. Figure 2 displays the equation conversion result of our running example. To conduct equation conversion, we design a recursive conversion algorithm based on the syntax tree structure. We first construct a quasi-binary syntactic tree for the original math equation. We denote the part which is on the left to the equal sign as f_1 (the formula in stage 1). For the part right to the equal sign, we build a binary tree with one operator op_1 as the root node which has two child trees. The child tree which has the new variable x' is marked as v_1 (the child tree with variable in stage 1) and the other one as n_1 (the child tree without variable in stage 1), and this identifying process is named as function *find_var* in Algorithm 1. This step corresponds to the upper left part of Figure 4. Next, like the upper middle part of Figure 4, we move n_1 to the left side and get a new operator op_1' according to equation reversion rules, which are summarized from the basic mathematical computation. We show two example

Rule	MWP examples	Num	Equation
1	A has 4 piles of 2 apples and B has 2 apples. B gave 1 apple to A, how many does A have in total now?	2	$x = 4 * 2 + 1$
2	The side length of a square is 2, what is the area?	2	$x = 2 * 2$
3	The side length of a cube is 4, what is the volume?	4 & 3	$x = 4^3$
4	The diameter of a circle is 5, what is the perimeter?	π	$x = \pi * 5$

Table 1: Examples of In-valid Numbers.

Algorithm 1: The algorithm of equation conversion

Input: Left part of medium equation l_tr and right part of equation r_tr

Output: Inverse equation in tree structure

```

1  $root = r\_tr.root$ ;
2  $num\_tr, var\_tr = find\_var(r\_tr)$ ;
3 while  $root \neq x'$  do
4    $l\_tree = rule(l\_tr, num\_tr, root)$ ;
5    $r\_tr = var\_tr$ ;
6    $root = r\_tr.root$ ;
7    $num\_tr, var\_tr = find\_var(r\_tr)$ 
8 return  $l\_tr = x'$ 

```

rules in Figure 5³. Regarding f_1 and n_1 as two new child trees and op'_1 as the new root node, we can get a new tree f_2 , which is the black circular ring in the upper right part of Figure 4, and we just use a single node f_2 to represent it in the following step.

Then, v_1 will be further analyzed as a binary tree which is composed of the root node op_2 and the child trees n_2 and v_2 as shown in the lower left part of Figure 4. If we ignore the dotted circular sign of v_1 , in this state the equation has the same structure as the beginning state, so the following process will repeat until v_n has only one node x' . At this time, referring to the lower right part of Figure 4, all nodes of numbers and operators are moved to the left side and only the variable node x' is left on the right side. The concrete process of equation conversion is shown in Algorithm 8.

Since one math equation can be written in several equivalent forms (e.g., $4 + 2 - 3$ and $4 - 3 + 2$), which bring noise to the model training, we conduct equation normalization for all the original and generated equations. We follow the criteria of Wang et al. (2018) that if one equation could be converted into a shorter one, then it should be shortened, and the order of the numbers in one equation should follow their occurrence order in the problem text as much as possible. Wang et al. (2018) performed equation normalization by applying a series of rules that only work on limited cases such as a sequence of multiplication. Here we increase the applicable scope. We also use the simplification algorithm supported by sympy (Meurer et al. 2017) that heuristically simplifies the equations and makes them match with human writing regulations. This equation normalization is essential for assuring the model performance since the equations gen-

erated by the reverse operation are written in a completely different style from that of the equations in the original data.

MWP Solving Model

Our data augmentation method can be combined with any preferred model. Here, we adopt two seq2seq models and examine the combination performance. First, we implement Liu et al. (2019)’s prefix model which is a light and effective baseline. At the same time, to show whether an advanced model can be further improved by our augmented data, we use the SOTA model named Goal-driven tree-structured MWP solver (GTS) (Xie and Sun 2019), which is an extension of the prefix baseline model. Here, we briefly introduce the two models.

For the prefix model, formally, the model takes a sequence of tokens $\{x_i\}_{i=0}^n$ as the input and embeds them into a sequence of word embedding representations $\{e_i\}_{i=0}^n$ which are fed into a bidirectional long short term memory network (BiLSTM) encoder. Then two context-aware representations $h_i^{enc_f}$ and $h_i^{enc_b}$ are calculated and concatenated as h_i^{enc} for each token. Then these representations are given to the decoder to decode the output equation.

The decoder adopts a unidirectional LSTM to generate the output in an autoregressive manner. At each decoding time step t , the decoder calculates the attention weight distribution $\{a_t^i\}$ on $\{h_i^{enc}\}$ with the embedding e_t^{dec} of the output of the previous time step y_{t-1} and the current hidden state h_t^{dec} of the decoder LSTM, and assigns them to the encoder outputs $\{h_i^{enc}\}$ to form an attention-aware representation s_t which is finally fed to a Multi-layer Perceptron (MLP) layer to generate the output token y_t .

$$h_t^d = LSTM(h_{t-1}^{dec}, e_t^{dec}) \quad (1)$$

$$s_t = \sum_{i=1}^n \alpha_t^i \cdot h_i^{enc} \quad (2)$$

$$= \sum_{i=1}^n \frac{\exp(h_i^{enc} \cdot h_t^{dec})}{\sum_{j=1}^n \exp(h_j^{enc} \cdot h_t^{dec})} \cdot h_i^{enc}$$

The GTS model (Xie and Sun 2019) further extends the prefix baseline with subtree representations which can provide more information for the decoding process. A recursive neural network is used to encode subtrees of the equation in a bottom-up manner. The subtree representation of one token y_t is calculated based on its children nodes with the gate mechanism. With the subtree representations, this model can also well use the information of the generated tokens to predict a new token.

³All the rules are listed in the appendix.

It is noted, for the seq2seq models, directly generating the solution equation with numbers suffer from a serious out-of-vocabulary (OOV) problem since the vocabulary of numbers is enormous. To address this problem, we follow Kushman et al. (2014), which used equation templates instead of actual equations as the prediction target of the model. The numbers in one MWP are notated as $temp_i$, where i denotes the order that the numbers appear in the problem. Extra constant numbers such as π and 1 are also added to the decoder vocabulary. Then the OOV problem can be solved.

To further improve the models, we mix and shuffle the augmented data with the original data as the new training set of the models. It is noticed that our augmented data is not limited to these two models.

Experiments

Experiment Setup

In our experiments, we mainly experiment on Math23K⁴ which is the most influential large scale dataset for MWP solving in the Chinese language. Math23k contains 23,162 elementary-school-level math word problems along with their equation solutions. The solution equation to each problem is linear and contains only one unknown variable. We do not use the large-scale dataset Dolphin18K (Shi et al. 2015), because it contains many small typos and wrong solutions, while our data augmentation method pursues producing high-quality new data from high-quality data. AQuA (Ling et al. 2017) and MathQA (Amini et al. 2019) use semantic parsing style output instead of equations so that our method is not applicable.

For the two seq2seq MWP solving models, we fix their embedding size to be 128. For the prefix baseline model, the dimension of encoder hidden state is set to 512 while the dimension of decoder hidden state is 1024. For the GTS model, the hidden state of both encoder and decoder is 512. We use Adam optimizer to optimize these parameters. The batch size is 128. To compare the performance with baselines fairly, GTS model is tested by 5-fold cross validation⁵, while others are tested on the test set⁶. The experiments are done on GTX 1080Ti GPU, with a runtime of 10 hours for prefix baseline and 110 hours for GTS.

For data augmentation, we perform reverse operation on the training set of Math23K which includes 21,162 MWPs. From these problems we can get 58,699 numbers through using the LSTM-based classifier. At the same time, we filter out 1,490 problems which are composed of only numbers and operators, because they can not give supervision to the MWP solving models. We also exclude 7,399 numbers in the problems which are not easy to be reversed, because they do not map one to one with the numbers in the solution equations as stated in Section 3.1. Finally we totally get 47,318

⁴Download link: https://github.com/SumbeeLei/Math_EN

⁵We filter out the augmented samples of test set for each cross. So in the training stage, models can only learn from the training set and their augmented samples.

⁶We use the same split as Wang et al. (2018). Only training set is used for augmentation.

Type	Count
Original Problems	21,162
Filtered Problems	1,490
Original Numbers	58,699
Candidate Numbers	54,717
Irreversible numbers	7,399
Augmented Problems	47,318

Table 2: The statistics of the data augmentation on the training set of Math23K.

new problem-equation pairs. We can see, our data augmentation method successfully generates the new data whose size is more than two times of the original data, demonstrating the method’s ability for performing large-scale data augmentation cheaply. We illustrate the statistics of the data augmentation results in Table 2.

MWP Solving Results

Model	Acc
Cosine* (2018)	23.8%
Jaccard* (2018)	47.2%
Self-Attention \diamond (2018)	56.8%
LSTM \diamond (2018)	57.9%
DNS \clubsuit (2017)	58.1%
Suf+EN \clubsuit (2018)	66.7%
TreeLSTM \clubsuit (2019)	69.0%
Group-Attention \clubsuit (2019)	69.5%
DNS+Retrieval \dagger (2017)	64.7%
Suf+EN Ensemble \dagger (2018)	68.4%
Prefix \clubsuit (2019)	67.8%
Prefix+RODA \clubsuit	70.5%
GTS \clubsuit (2019)	74.3%
GTS+RODA \clubsuit	76.0%

Table 3: Math word problem solving accuracy on Math23K. * denotes retrieval model, \diamond denotes classification model, \clubsuit means the generation model. \dagger denotes the model ensemble.

We first evaluate the MWP solving performance by comparing our methods with other baseline methods. Here we name our reverse operation based data augmentation method as RODA. In this experiment, we use all the 47,318 augmented data for training the prefix and GTS models. Table 3 shows the results of our methods and other novel systems on the Math23k evaluated by the final answer accuracy. In this table, we classify the MWP solving models as retrieval-based, classification-based, generation-based and ensemble models. The retrieval-based models mainly calculates a similarity score for questions in the test set and the questions in the training set and assign the template that has the highest similarity (Upadhyay and Chang 2017; Robaidek, Koncel-Kedziorski, and Hajishirzi 2018). **Cosine** and **Jaccard** respectively denote the methods which adopt the corresponding similarities. The classification-based models train a classifier to predict an equa-

Data	Data Prop.	Acc.
Original only	–	68.0%
RODA Only(2.24 times)	–	50.0%
Orig+RODA	1:0.5	69.5%
Orig+RODA	1:1	69.8 %
Orig+RODA	1:1.5	70.9 %
Orig+RODA(All)	1:2.24	71.0%

Table 4: Effects of data augmentation by adjusting the augmentation proportion on the development set. The middle column denotes the proportion of original data and augmented data.

tion template for each problem in a multi-class classification manner (Kushman et al. 2014). For retrieval and classification models, we use the results from Robaidek, Koncel-Kedziorski, and Hajishirzi (2018).

The generation-based models are the recent mainstream for MWP solving and use end-to-end seq2seq models to directly generate an equation template. Wang, Liu, and Shi (2017) proposed the **DNS** model, which used seq2seq with significant number identification to generate an equation template. Wang et al. (2018) improved their model and proposed the **Suf+EN** model, which extends the DNS model by decoding the suffix notation and performs equation normalization for preprocessing. **TreeLSTM** (Liu et al. 2019) uses a top-down tree-structured decoder to predict the equations. **Group-Attention** (Li et al. 2019) uses various attention methods to capture the intra-relation of the numbers. We also use two ensemble models **DNS+Retrieval** and **Suf+EN Ensemble** for comparison, which use bagging to combine the results of different models.

Our data augmentation method is exerted on two generation models Prefix Liu et al. (2019) and GTS (Xie and Sun 2019) which have been introduced in Subsection 3.2. We choose these models as they can somewhat be representative of generation-based methods especially GTS achieves the SOTA performance. From Table 3, we can see that generation-based methods generally outperform retrieval-based and classification-based methods. Our RODA method can further promote the current SOTA models: Prefix+RODA and GTS+RODA boost Prefix and GTS about 2.7 points and 1.7 points respectively. This also exhibits that more high-quality data is useful for improving the performance of MWP solving.

Analysis of Data Augmentation

Here we further investigate how the augmented data improves the MWP solving model. In consideration of balance of experiment time and accuracy, we use the Prefix model on the development set of Math23K for analysis.

Previous studies on data augmentation (Yu et al. 2018) show that too much augmented data might harm the performance of the model. Thus, we experiment what percentage of our augmented data can best improve the MWP solving model. We only use our augmented data to train the model and achieve the accuracy of 50%, still far lower than only using the original Math23k training data (68%). This

Model	Acc
BT Only	45.6 %
RODA(1:1) Only	48.2%
RODA(1:2.24) Only	50.0%
Origin	68.0%
+ BT	68.2 %
+ RODA(1:1)	69.8%
+ Full RODA	71.0%

Table 5: Comparison with Back Translate data augmentation method.

means that the quality of our augmented data can be further improved. We also consider combine the original data with different proportions of augmentation data as training data whose performance is shown in Table 4. We can see that, as the size of the augmented data increases, the performance stably increases, which shows how the size of the training data effects the performance. The model performs similar when the proportion is 1.5 and using the whole augmented data. Because the performance of the model has not decreased along with the increase of augmented data, we use the full augmented data for the reported results. This can demonstrate the reverse operation can infer new knowledge points which give supervision for new mathematical reasoning logic while more high-quality data can be used to well train the neural networks.

We also compare our data augmentation method with another novel data augmentation method back-translate (BT) (Yu et al. 2018) . For the BT method, we translate the problem text into English and then back to Chinese by Google Translate⁷. As BT can only performs data augmentation with the 1:1 proportion of the original data, we also control the size of our augmented data. Table 5 compares the MWP solving results. As shown in Table 5, We can see that the performance of BT has a significant gap with RODA, even when the data augmentation proportion is the same. There are two reasons for such performance gap. First BT may introduces more noise into the training data through the translation-based paraphrase and degrade the model performance, while our method can better ensure data quality. Second BT only paraphrases the same meaning on the lexicon level, while our data augmentation method can introduce new knowledge points that leads to new mathematical reasoning logic.

Human Evaluation

To further examine the data quality of our augmented data, we perform human evaluation to examine the original data, back-translate augmented data and RODA data. We sample 100 MWPs from each dataset. The evaluation involves two aspects. The first is the coherence which is ranked between 1-5. This examines whether the generated text is coherent. The second is correctness which is classified either 0 or 1. This examines whether the equation matches with the problem text. Two annotators participate in ranking the data. Ta-

⁷<https://translate.google.com/>

#	Chinese Text	English Translation	Equation	N	C
1	甲、乙两同学相距 t_a 米,同时相向而行,乙同学速度为 t_b 米/秒,与此同时,一只小狗以 t_c 米/秒的速度,从甲身边跑向乙,遇到乙后又以同样的速度跑向甲,...如此往返,直到甲、乙同学相遇,问在此段时间内,小狗共跑了 t_d 米,甲同学速度为多少米/秒?	The distance between A and B is t_a meters. The two are heading toward each other. The speed of B is t_b m/s. Meantime, a dog starts with A and then runs back and forth between the two people with a speed of t_c m/s, until the two people meet. During this duration, the dog ran t_d meters. What is the speed of A?	$x = t_a * t_c / t_d - t_b$	5	1
2	几个小朋友分苹果,如果每人分 t_a 个,如果每人分 t_b 个,少 t_c 个,小朋友有 t_d 人,就余多少个?	A few children are sharing apples. If each child gets t_a apples. If each child gets t_b apples, there would be a shortage of t_c apples. There are t_d children. How many apples are left?	$x = t_d * (t_b - t_a) - t_c$	2	0

Table 6: Examples of the output from the Data Augmentation Module.

ble 7 lists the average scores for each dataset with respect to the two metrics. From Table 7, we can see our method can generate new data which has only a small performance gap with the original data. Compared to the BT method, our augmented data is higher quality in both coherence and correctness. This can demonstrate how our method is reliable so that the key information is not lost, which is also why our augmented data can well boost the model performance.

Model	Coherence	Correctness
Original	4.27	0.92
BT	3.24	0.55
RODA	3.86	0.84

Table 7: Human evaluation on Math23K.

Here we show two augmented example in Table 9. In case 1, this newly produced example has coherent text and correct solution, even if the original mathematical logic and description text is fairly complex. The original example involves the numbers that the time used by A, B and the dog is the same, so that it can form the problems for each of the 5 variables: the speed of A, B and the dog, the distance between A and B and the distance that the dog has run. In this example the original training data can be augmented into 4 new high quality question-answer pairs, and each of them holds a new mathematical reasoning logic, which demonstrates the effectiveness of our model. We also show an example where our method failed in case 2. When swapping the order of the sentences, the coreference resolution of **apples** in the final question has changed that it no longer asks about the origin variable, therefore the new reversed question would no longer be natural nor correct. In the case that the sentence order swapping would influence the coreference resolution of natural text, our method would no longer work. Such kind of error could be reduced with an additional discourse analysis module, which could be left for further work.

Model	Acc
GTS	68.2%
+ BT	70.3 %
+ Full RODA	70.7%

Table 8: Results on AllArith.

A Study on English Dataset

We also extend our data augmentation method to AllArith⁸ (Roy and Roth 2017), which is a high-quality English MWP dataset with 831 problems, to show how our method works on an English dataset. Our data augmentation method generated 715 new examples for AllArith, respectively. We use the GTS model in this experiment and compare with the BT data augmentation method. The results are based on 5-cross validation following the split of Roy and Roth (2017). For problem transformation, we follow the manually encoded transformation rules from Heilman and Smith (2009). As we can see in Table 8, our model achieves performance improvement when adding the augmented data, which demonstrates the generalization ability of our method beyond language. Here the BT data also achieves performance improvement, slightly lower than using our augmented data. Since the size of AllArith is small and its examples are simpler than Math23K in mathematical reasoning complexity, our data augmentation method does not exhibit significant advantages, because our method is also restricted to the scale of the original data. In future, we will research how to produce more high-quality data.

Conclusion

In this paper, we propose the reverse operation based data augmentation for MWP solving, which converts the question and equation via reverse operation. Enlightened by how human perform double-checking during calculation, the method can perform cheap and accurate data augmentation that could be adapted to any model. The augmented data also provides supervision of a new mathematical reasoning logic that could benefit the model beyond paraphrasing the text.

⁸Download link: <https://github.com/CogComp/arithmetic>

We evaluate our method on Math23K and achieve state-of-the-art performance.

References

- Amini, A.; Gabriel, S.; Lin, S.; Koncel-Kedziorski, R.; Choi, Y.; and Hajishirzi, H. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2357–2367. Minneapolis, Minnesota.
- Asai, A.; and Hajishirzi, H. 2020. Logic-Guided Data Augmentation and Regularization for Consistent Question Answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5642–5650. Association for Computational Linguistics.
- Bobrow, D. G. 1964. Natural Language Input for a Computer Problem Solving System. Technical report, Cambridge, MA, USA.
- Charniak, E. 1969. Computer Solution of Calculus Word Problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence, IJCAI’69*, 303–316. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL <http://dl.acm.org/citation.cfm?id=1624562.1624593>.
- Chiang, T.-R.; and Chen, Y.-N. 2018. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems. *arXiv preprint arXiv:1811.00720*.
- Dua, D.; Wang, Y.; Dasigi, P.; Stanovsky, G.; Singh, S.; and Gardner, M. 2019. DROP: A Reading Comprehension Benchmark Requiring Discrete Reasoning Over Paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2368–2378.
- Gokhale, T.; Banerjee, P.; Baral, C.; and Yang, Y. 2020. VQA-LOL: Visual question answering under the lens of logic. *arXiv preprint arXiv:2002.08325*.
- Guan, W.; Liu, Q.; Han, G.; Wang, B.; and Li, S. 2019. An Improved Coarse-to-Fine Method for Solving Generation Tasks. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, 178–185. Sydney, Australia: Australasian Language Technology Association. URL <https://www.aclweb.org/anthology/U19-1024>.
- Heilman, M.; and Smith, N. A. 2009. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST.
- Kang, D.; Khot, T.; Sabharwal, A.; and Hovy, E. 2018. AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2418–2428.
- Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 271–281.
- 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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6162–6167.
- Ling, W.; Yogatama, D.; Dyer, C.; and Blunsom, P. 2017. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 158–167.
- Liu, Q.; Guan, W.; Li, S.; and Kawahara, D. 2019. 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)*, 2370–2379.
- Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3: e103. ISSN 2376-5992. doi:10.7717/peerj-cs.103. URL <https://doi.org/10.7717/peerj-cs.103>.
- Nie, F.; Wang, J.; Yao, J.-G.; Pan, R.; and Lin, C.-Y. 2018. Operation-guided Neural Networks for High Fidelity Data-To-Text Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3879–3889. Brussels, Belgium: Association for Computational Linguistics.
- Robaidek, B.; Koncel-Kedziorski, R.; and Hajishirzi, H. 2018. Data-Driven Methods for Solving Algebra Word Problems. *arXiv preprint arXiv:1804.10718*.
- Roy, S.; and Roth, D. 2015. Solving General Arithmetic Word Problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1743–1752.
- Roy, S.; and Roth, D. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- Roy, S.; and Roth, D. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics* 6: 159–172.
- Shi, S.; Wang, Y.; Lin, C.-Y.; Liu, X.; and Rui, Y. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1132–1142.

Upadhyay, S.; and Chang, M.-W. 2017. Annotating Derivations: A New Evaluation Strategy and Dataset for Algebra Word Problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 494–504.

Wang, L.; Wang, Y.; Cai, D.; Zhang, D.; and Liu, X. 2018. Translating a Math Word Problem to a Expression Tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 1064–1069.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B.; and Shen, H. T. 2019. Template-Based Math Word Problem Solvers with Recursive Neural Networks .

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854.

Wei, J.; and Zou, K. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. 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)*, 6381–6387. Hong Kong, China: Association for Computational Linguistics.

Xie, Z.; and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 5299–5305. doi:10.24963/ijcai.2019/736. URL <https://doi.org/10.24963/ijcai.2019/736>.

Yu, A. W.; Dohan, D.; Luong, M.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

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 Bessiere, C., ed., *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 4011–4017. International Joint Conferences on Artificial Intelligence Organization. Main track.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3928–3937. Online: Association for Computational Linguistics.

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

Appedices

Original Examples

We show some example generated by the data augmentation module in Chinese in Table 9. We can see even when the text and mathematical logic is very complex which involves with seven numbers, our algorithm can produce high quality data.

Interrogative Pronouns

We show the list of interrogative pronouns used for question conversion here in Table 10. Some of the interrogative pronouns also have other meanings in the declarative discourse unit, so we only detect them if they are in the last discourse unit of the sentence.

Removing Negative

To be noticed that sympy would put negative numbers in the first position of one equation during simplification, which causes the problem that one equation no longer can form a binary abstract syntax tree. We also design an algorithm to avoid this problem by moving the first addition term to the first position. We show an example in Table 11.

This equation normalization is essential for our algorithm since the equations generated by the reverse operation is written in a style completely different from the equations in the original data. The normalization can avoid such domain shift.

We show the algorithm for removing the negative terms in the front of the equation during equation normalization in Algorithm 2. To be noticed, this process does not change the meaning of the equation but only how it is written.

Algorithm 2: Remove Negative

Result: New Equation Written Form

```

9 Input Equation while True do
10   instructions;
11   if Brackets in Equation then
12     subs = list of sub-equation in the brackets;
13     for sub-equation in subs do
14       Remove Negative(sub-equation);
15   else
16     if Equation[0]== '-' then
17       Find the first add token;
18       Move the token to the front;
```

Inverse Rules

The full set of rules of inverse operation is shown in Figure 5.

Implementation Details

We follow Liu et al. (2019) and Xie and Sun (2019) for all of the hyperparameters.

#	Chinese Text	Equation	Natural	Correct
1	果园里有 $temp_a$ 棵苹果树, 苹果树比桃树多 $temp_b$ 棵, 是桃树的多少倍?	$x = temp_a / (temp_a - temp_b)$	5	1
2	杨师傅 $temp_a$ 小时生产零件 $temp_b$ 个, 技术革新后, 生产效率提高了百分之 $temp_c$, 几小时生产 $temp_d$ 个?	$x = \frac{temp_d}{temp_b * 5 + temp_c / temp_a + temp_b / temp_a}$	5	1
3	$temp_a$ 名少先队员乘坐 $temp_b$ 辆大客车去秋游, 最后一辆车要坐 $temp_c$ 个同学, 前多少辆车每辆乘坐 $temp_d$ 人?	$x = (temp_a - temp_c) / temp_d$	4	1
4	五年级同学向希望小学捐款, 第一小队 $temp_a$ 人, 共捐款 $temp_b$ 元; 第三小队 $temp_c$ 人, 共捐款 $temp_d$ 元, 全班平均每人捐款 $temp_e$ 元, 共捐款 $temp_f$ 元; 第二小队多少人?	$x = \frac{(temp_b + temp_d + temp_f) / temp_e - temp_a - temp_c}{temp_e}$	5	1

Table 9: Examples of the output from the Data Augmentation Module.

Chinese	English Translation
多少	How many/much
几分之几	the fraction number is
几	How many
=†	=
求	Please solve
((()/((/)))	((()/((/)))
多†	how much more than

Table 10: A list of interrogative pronouns. † denotes this pronoun is only valid when it is in the last discourse unit of the question..

Step	Equation
Equation	$x = c - a - c + (c * a) + (b/b)$
Sympy	$x = -a + 1 + (a * c)$
Adjusted	$x = 1 - a + (a * c)$

Table 11: One example of equation normalization

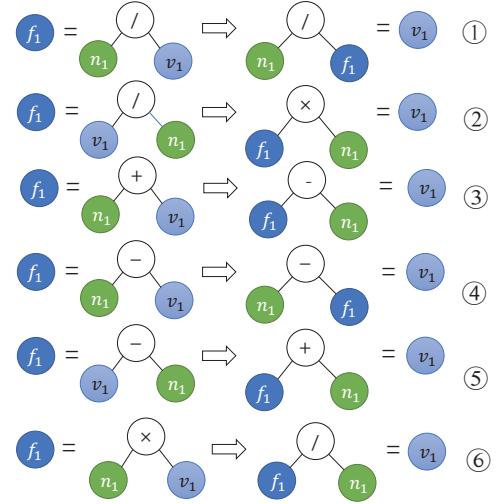


Figure 5: Two rules of inverse operation

Reverse Operation based Data Augmentation for Solving Math Word Problems

Anonymous Submission

Appedices

Interrogative Pronouns

We show the list of interrogative pronouns used for question conversion here in Table 1. Some of the interrogative pronouns also have other meanings in the declarative discourse unit, so we only detect them if they are in the last discourse unit of the sentence.

Chinese	English Translation
多少	How many/much
几分之几	the fraction number is
几	How many
=†	=
求	Please solve
((O))/(O))	((O))/(O))
多†	how much more than

Table 1: A list of interrogative pronouns. † denotes this pronoun is only valid when it is in the last discourse unit of the question..

Removing Negative

To be noticed that sympy would put negative numbers in the first position of one equation during simplification, which causes the problem that one equation no longer can form a binary abstract syntax tree. We also design an algorithm to avoid this problem by moving the first addition term to the first position. We show an example in Table 2.

This equation normalization is essential for our algorithm since the equations generated by the reverse operation is written in a style completely different from the equations in the original data. The normalization can avoid such domain shift.

We show the algorithm for removing the negative terms in the front of the equation during equation normalization in Algorithm 1. To be noticed, this process does not change the meaning of the equation but only how it is written.

Step	Equation
Equation	$x = c - a - c + (c * a) + (b/b)$
Sympy	$x = -a + 1 + (a * c)$
Adjusted	$x = 1 - a + (a * c)$

Table 2: One example of equation normalization

Algorithm 1: Remove Negative

Result: New Equation Written Form

```

1 Input Equation while True do
2   instructions;
3   if Brackets in Equation then
4     subs = list of sub-equation in the brackets;
5     for sub-equation in subs do
6       Remove Negative(sub-equation);
7   else
8     if Equation[0]='-' then
9       Find the first add token;
10      Move the token to the front;
```

Original Examples

We show some example generaed by the data augmentation module in Chinese. We can see even when the text and mathematical logic is very complex which involves with seven numbers, our algorithm can produce high quality data.

Inverse Rules

The full set of rules of inverse operation is shown in Figure 1.

Implementation Details

We follow ? and ? for all of the hyperparameters.

Hard and Easy examples

We study how the difficulty of the augmented data effects the performance of the model in Table 4, by only using 'hard' or 'easy' examples during training. The difficult is simply identified by how many numbers are used in the equation. Using 4 or more numbers would be considered as a hard example and using 3 or less numbers would be considered as an easy example. The results show that the easy examples

#	Chinese Text	Equation	Natural	Correct
1	果园里有 $temp_a$ 棵苹果树, 苹果树比桃树多 $temp_b$ 棵, 是桃树的多少倍?	$x = temp_a / (temp_a - temp_b)$	5	1
2	杨师傅 $temp_a$ 小时生产零件 $temp_b$ 个, 技术革新后, 生产效率提高了百分之 $temp_c$, 几小时生产 $temp_d$ 个?	$x = \frac{temp_d}{temp_c / temp_a + temp_b / temp_a} * 5$	5	1
3	$temp_a$ 名少先队员乘坐 $temp_b$ 辆大客车去秋游, 最后一辆车要坐 $temp_c$ 个同学, 前多少辆车每辆乘坐 $temp_d$ 人?	$x = (temp_a - temp_c) / temp_d$	4	1
4	五年级同学向希望小学捐款, 第一小队 $temp_a$ 人, 共捐款 $temp_b$ 元; 第三小队 $temp_c$ 人, 共捐款 $temp_d$ 元, 全班平均每人捐款 $temp_e$ 元, 共捐款 $temp_f$ 元; 第二小队多少人?	$x = \frac{(temp_b + temp_d + temp_f) / temp_e - temp_a - temp_c}{temp_e - temp_a - temp_c}$	5	1

Table 3: Examples of the output from the Data Augmentation Module.

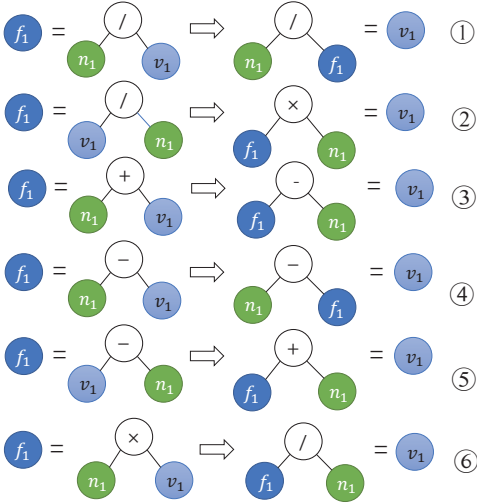


Figure 1: Two rules of inverse operation

are more effective. This could be a result of the size of the data, since there are much more easy examples than hard examples.

Model	Acc
Prefix Baseline	68.0%
+ Hard Examples	69.5 %
+ Easy Examples	70.3 %
Full Augmentation	71.0%

Table 4: Ablation study on Math23K by only using hard or easy examples.