MWP-BERT: Numeracy-Augmented Pre-training for Math Word Problem Solving

Zhenwen Liang[1], **Jipeng Zhang**[2], **Lei Wang**[3],
Wei Qin[4], **Yunshi Lan**[5], **Jie Shao**[6], and **Xiangliang Zhang**[✉1]

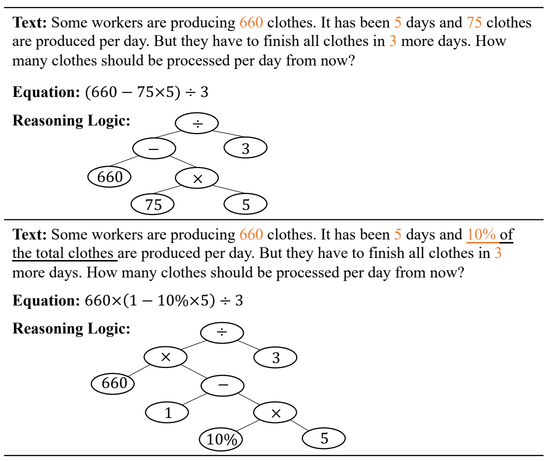
[1]University of Notre Dame, {zliang6, xzhang33}@nd.edu
[2]Hong Kong University of Science and Technology, jzhanggr@conect.ust.hk
[3]Singapore Management University, lei.wang.2019@phdcs.smu.edu.sg
[4]Hefei University of Technology, qinwei.hfut@gmail.com
[5]East China Normal University, yslan@dase.ecnu.edu.cn
[6]University of Electronic Science and Technology of China, shaojie@uestc.edu.cn

Abstract

Math word problem (MWP) solving faces a dilemma in number representation learning. In order to avoid the number representation issue and reduce the search space of feasible solutions, existing works striving for MWP solving usually replace real numbers with symbolic placeholders to focus on logic reasoning. However, different from common symbolic reasoning tasks like program synthesis and knowledge graph reasoning, MWP solving has extra requirements in numerical reasoning. In other words, instead of the number value itself, it is the reusable numerical property that matters more in numerical reasoning. Therefore, we argue that injecting numerical properties into symbolic placeholders with contextualized representation learning schema can provide a way out of the dilemma in the number representation issue here. In this work, we introduce this idea to the popular pre-training language model (PLM) techniques and build MWP-BERT, an effective contextual number representation PLM. We demonstrate the effectiveness of our MWP-BERT on MWP solving and several MWP-specific understanding tasks on both English and Chinese benchmarks.

1 Introduction

Recent works in math word problem (MWP) solving (Wang et al., 2018, 2019; Liu et al., 2019a; Li et al., 2019; Xie and Sun, 2019; Zhang et al., 2020b; Wu et al., 2020; Qin et al., 2021; Huang et al., 2021; Wu et al., 2021a; Yu et al., 2021; Shen et al., 2021) arrange the pipeline into a sequence-to-sequence framework. In brief, they use deep representation and gradient optimization as well as symbolic constraints to discover discrete symbolic combinations of operators and variants. Fundamentally, MWP solving system aims to perform symbolic reasoning by searching through a combinatorial solution space given the text description


Text: Some workers are producing 660 clothes. It has been 5 days and 75 clothes are produced per day. But they have to finish all clothes in 3 more days. How many clothes should be processed per day from now?

Equation: $(660 - 75 \times 5) \div 3$

Reasoning Logic:

Text: Some workers are producing 660 clothes. It has been 5 days and 10% of the total clothes are produced per day. But they have to finish all clothes in 3 more days. How many clothes should be processed per day from now?

Equation: $660 \times (1 - 10\% \times 5) \div 3$

Reasoning Logic:

Figure 1: The second question is obtained from the first one by minor modifications. However, their solution equation and corresponding equation tree structure are different from each other. This demonstrates the importance of considering numerical value information and reasoning logic (equation tree) in contextual modeling.

evidences. Thus, these neurosymbolic methods mainly focus on getting more effective semantic representations (Li et al., 2019; Zhang et al., 2020b; Wu et al., 2020, 2021a; Yu et al., 2021), injecting symbolic constraints (Wang et al., 2018, 2019; Liu et al., 2019a; Xie and Sun, 2019) and how to align semantic space (text descriptions) and huge combinatorial symbolic space (symbolic solutions) (Qin et al., 2021; Shen et al., 2021; Huang et al., 2021). This line of methods has achieved great success and is still holding the lead in various MWP solving benchmarks (Wang et al., 2017; Zhao et al., 2020; Koncel-Kedziorski et al., 2016).

Despite the great performance achieved by the previous methods, there still exists fundamental challenges in number representation for MWP solving. More exactly, number values are required to be considered as vital evidence in solution exploration but existing works are known to be inefficient in capturing numeracy information (Wallace et al., 2019). Intuitively, we could simply treat explicit numbers in the same way with words, i.e., assign position for all numbers in the vocabulary. However, there would be an infinite number of candidates during prediction and it would be impossible to learn their deep representations. In other words, the solution space will be extremely large and the complexity is unacceptable. Therefore, almost all existing works follow the number mapping technique Wang et al. (2017) to replace all numbers with symbolic placeholders (e.g., "x1", "x2"). The core idea here is to get a reasonable solution space by restricting neural networks to leave out numerical characteristics and focus on logic reasoning. However, most of the current MWP solvers do not consider the background knowledge in the context and are usually inefficient in capturing numeracy properties. An example is shown in Fig. 1. Small perturbations in the problem description actually bring large variations in reasoning logic and equation. If the model simply regards "75" and "10%" as the same placeholder "x3", and does not notice the small variation in the context, a wrong solution will be generated.

To this end, we incorporate several numeracy grounded pre-training objectives to inject inductive bias about numerical constraints into dynamic representations. Compared with word candidate sets, useful points in number candidate space are scattered sparsely. However, we identify that during prediction, what matters is the reusable numerical properties of number values. What's more, these properties do not suffer from the sparsity issues of specific values in MWPs. Therefore, compared with assigning a prototype vector for each single number value like (Wu et al., 2021b), it is more reasonable to inject the reusable numerical properties in deep representations, e.g., magnitude and number type. In this work, we propose to design numeracy grounded pre-training objectives to implement soft constraints between symbolic placeholders and numbers in deep representation.

Contributions. We present a suite of numeracy-augmented pre-training tasks with consideration of reasoning logic and numerical properties. More exactly, we introduce several novel pre-training tasks with access to different levels of supervision signals to make use of more available MWP data.

- One basic group of pre-training tasks is designed for the self-supervised setting. Except for masked language modeling (MLM), we give extra consideration to number-related context information by designing related objectives.
- Another set of pre-training objectives is for the weakly-supervised setting that has only answer annotations but no equation solutions. With access to the answer value, we introduce several tasks to determine the type and the value of the answer.
- The final set is for the fully-supervised setting, where both solution equation and answer are available for the MWPs.

Besides, a group of numeracy grounded pre-training objectives is designed to leverage the corpus of MWP and encourage the contextual representation to capture numerical information. Experiments conducted on both Chinese and English benchmarks show the significant improvement of our proposed approach over all competitors. To our knowledge, this is the first approach that surpasses human performance (Wang et al., 2019) in terms of MWP solving.

2 Related Works

Math Word Problems Solving. There exist two major types of MWP, equation set MWP (Wang et al., 2017; Zhao et al., 2020) and arithmetic MWP (Qin et al., 2020; Huang et al., 2016). This work focuses on arithmetic MWP, which is usually paired with one unknown variable. Along the path of the MWP solver's development, the pioneer studies use traditional rule-based methods, machine learning methods and statistical methods (Yuhui et al., 2010; Kushman et al., 2014; Shi et al., 2015; Koncel-Kedziorski et al., 2015). Afterwards, inspired by the development of sequence-to-sequence (Seq2Seq) models, MWP solving has been formulated as a neurosymbolic reasoning pipeline of translating language descriptions to mathematical equations with encoder-decoder framework (Wang et al., 2018, 2019; Li et al., 2019; Zhang et al., 2020b; Yu et al., 2021; Wu et al., 2021a). By fusing hard constraints into decoder (Chiang and Chen, 2018; Liu et al., 2019a; Xie and Sun, 2019; Shen and Jin, 2020; Zhang et al., 2020a), MWP solvers achieve much better performance then. Several works propose to utilize multi-stage frameworks (Wang et al., 2019; Huang et al., 2021; Shen et al., 2021; Liang and Zhang, 2021) to make more robust solvers. Also, several new works made attempts to improve MWP solver beyond supervised settings (Hong et al., 2021a,b).

Among all these previous studies, the most relevant ones to our work can be categorized into two groups. First, it has been noted that number values and mathematical constraints play a significant role in supporting numerical reasoning. Wu et al. (2021b) proposed several number value features to enhance encoder and Qin et al. (2021) designed new auxiliary tasks to enhance neural MWP solvers. Compared with their work, we first introduce pre-training language model (PLM) and concentrate on representation learning to resolve numerical understanding challenges. Second, regarding the usage of pre-training techniques for MWP solving, Shen et al. (2021) introduced BART-based (Lewis et al., 2020) MWP solver and incorporated specialized multi-task training for obtaining more effective pre-training Seq2Seq models for MWP. Compared with them, our work focuses on the number representation learning issue of MWP and achieves a more flexible pre-training representation module for MWP solving, which can be applied in various MWP related tasks other than solution generation.

Numeracy-aware Pre-training Models. Number representation has been recognized as one of the main issues in word representation learning. Existing methods make use of value, exponent, subword and character methods (Thawani et al., 2021) to obtain number representations for explicit number values. These methods are known to be less effective in extrapolation cases like testing with numbers not appearing in the training corpus.

Previous related works (Andor et al., 2019; Wallace et al., 2019; Geva et al., 2020) mainly focus on shallow numerical reasoning tasks shown in DROP dataset (Dua et al., 2019), which usually serves as a benchmark for evaluating numerical machine reading comprehension (Num-MRC) performance. Compared with MWP solving, Num-MRC's main focus is laid on extracting answer spans from a paragraph, which are more fault-tolerant with no needs to predict number tokens. Besides, their solution generation tasks only contain simple computations like addition/subtraction and there are only integers in DROP. More exactly, several research efforts have been made to deal with this kind of math-related reading comprehension task by synthesizing new training examples (Geva et al., 2020),

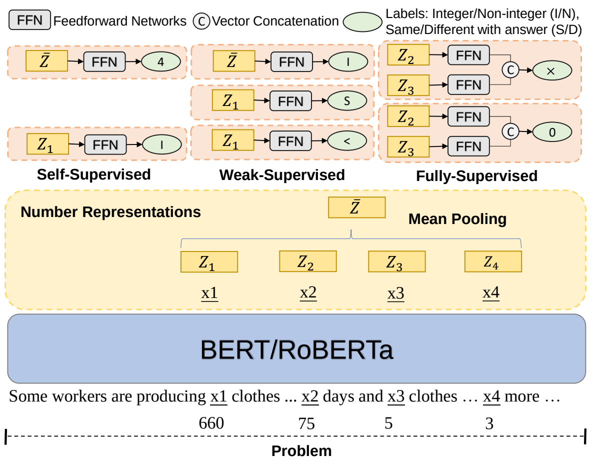


Figure 2: The overall architecture of our BERT-based MWP solver. Our method enables the solver to learn from unlabeled, incompletely labeled and fully labeled MWPs by different pre-training tasks.

incorporating special modules considering the numerical operation (Andor et al., 2019) and designing specific tokenization strategies (Zhang et al., 2020c). Since MWP solving requires further consideration of the complex composition of reasoning logics in MWP text, the symbolic placeholder is more effective in MWP solving. Thus, instead of dealing with explicit number values, our work focuses on improving representation for symbolic placeholders by injecting numerical properties in a probabilistic way.

3 Methodology

3.1 Problem Statement

The input to an MWP solver is a textual description, we denote it as W with length m, thus $W = \{w_1, w_2, ..., w_m\}$. We also define a subset W_q of W which contains all the quantities appeared in W. The output is an equation showing how to get the final answer. We denote it as A with length n, where $A = \{a_1, a_2, ..., a_n\}$. The vocabulary of A contains three parts, namely V_{op}, V_{num} and V_{cons}. V_{op} is the vocabulary for all operators, i.e. $+$, $-$, $\times$, $\div$ and ^. The vocabulary of quantities V_{num} is constructed by number mapping (Wang et al., 2017), which transforms quantities in different MWPs into a unified representation. More specifically, V_{num} does not contain the actual value of quantities appeared in W, and those quantities are denoted as $\{n_1, n_2, ..., n_k\}$, where n_i means the i-th number from W and k is the maximum number of quantities in V_{num} in order to fix the size of it. V_{cons} contains necessary constant values e.g., π.

3.2 PLM Encoder

Our PLM encoder maps the problem description W into a representation matrix $Z \in \mathbb{R}^{m*h}$ where h is the dimension of the hidden feature.

$$Z = encoder(W). \tag{1}$$

The representation vector corresponding to each word in Z will be used in the decoding process for generating the solution.

An overview of pre-training objectives and our model architecture is shown in Figure 2. In general, pre-training objectives are designed to inject contextual priori and numerical properties as soft constraints for representation learning. They are categorized into three types given provided training signals, i.e., self-supervised, weakly-supervised, and fully-supervised.

3.3 Self-supervised Objectives

In this part, we only consider input text descriptions for each example. Also, these objectives can alleviate the costs of collecting MWP corpus by constructing supervision signals without solution answers and equations.

Masked Language Modeling. We follow Devlin et al. (2019) and introduce masked language modeling (MLM) for basic contextual representation modeling. Specially, we apply masks on 10% of tokens, randomly replace 10% of tokens with other tokens and keep 80% of tokens unchanged. Later, the manipulated sentence is utilized to reconstruct the original sentence.

Number Counting. Another pre-training objective is to predict the amount of numbers that appeared in MWP description. The amount of a number corresponds with the cardinality of variable sets. This also reflects the basic understanding about the difficulty of an MWP and can act as a key contextual MWP number understanding feature. Here, we introduce a regression objective with the following formulated loss function:

$$L_{NumCount} = MSE(FFN(\bar{Z}), |W_q|), \tag{2}$$

where MSE stands for mean-squared-error and FFN stands for the feed-forward network which is made up of two fully-connected (FC) layers and one ReLU activation. We build a two-FC-layer block for each pre-training task (except MLM) and discard them during fine-tuning. $\bar{Z} \in \mathbb{R}^h$ is the mean vector of Z and represents the encoder's overall understanding of a single MWP text description. $|W_q|$ is the number of quantities shown in the problem description W.

Number Type Grounding. This objective aims at linking contextual number representations with corresponding number types to tell the difference between discrete and continuous concepts/entities. For numerical reasoning in MWP solving, we only need to handle whole numbers as well as non-integer numbers (decimal, fraction and percentage). Ideas here are that whole numbers usually associate with discrete entities (for example, desks, chairs and seats) while non-integer numbers often connect with continuous concepts (for example, proportions, rate, velocity). Besides, comparisons among whole numbers got different issues compared with rational numbers. Therefore, we propose a classification objective to predict if a number is a whole number or non-integer number:

$$L_{NTGround} = \sum_{i:W_i \in W_q} CE(FFN(Z_i), y_i), \tag{3}$$

where W_q contains all the numbers that appeared in W, and CE is the cross-entropy loss for binary classification. Here, i is the index when W_i is a quantity, Z_i is the corresponding representation vector, and y_i is a binary label indicating if W_i is a whole number or non-integer number.

3.4 Weakly-supervised Objectives

Given both text descriptions of MWPs and corresponding answers, we can model dependencies among answer number and numbers in text descriptions so that contextual representation perceive the existence of the target variable number that does not appear in the text descriptions. In detail, we design 3 novel pre-training objectives specializing in value-annotated MWPs to improve number representation in our MWP-BERT.

Answer Type Prediction. Determining the type of answer number can provide us discrete/continuous nature of target entity/concept. Thus, we want to predict type (whole/non-integer) of the answer value given global representations of an MWP (embedded in Z):

$$L_{ATPred} = CE(FFN(\bar{Z}), y_s), \tag{4}$$

where y_s is the ground truth label indicating the type of answer number.

Context-Answer Type Comparison. Besides the global context feature, an MWP-BERT also needs to associate context numbers and answer number (the target number does not explicitly appear in the text). Thus, another objective is proposed to predict if the quantities appeared in the MWP text fall into the same category as the answer (i.e. they are all whole or non-integer):

$$L_{CATComp} = \sum_{i:W_i \in W_q} CE(FFN(Z_i), y_i \oplus y_s), \tag{5}$$

where $\oplus$ stands for the exclusive-or operator between two binary labels to check if they are the same, the label of a quantity y_i and the label of the solution value y_s.

Number Magnitude Comparison. Beyond type, the magnitude of a number serves as the foundation of numerical reasoning. By associating magnitudes evaluation with contextual representation, the model can get a better perception about variance over key reasoning cues like time, size, intensity and speed. Let $\dot{y}_i$ indicate if the current quantity W_i is greater than the solution value or not. Moreover, the loss function is formulated as:

$$L_{NumMComp} = \sum_{i:W_i \in W_q} CE(FFN(Z_i), \dot{y}_i). \tag{6}$$

3.5 Fully-supervised Objectives

Given both equations and answers for MWPs, we can design fully-supervised training tasks to associate number representation with reasoning flows (solution equation). Mathematical equations are known to be binary tree structures with operators on root nodes and numbers on leaf nodes. The motivation is to encourage models to learn structure-aware number representations that encode the information on how to make combinations over atomic operators and numbers. We incorporate two pre-training objectives based on the solution equation tree.

Operation Prediction. The first one is a quantity-pair relation prediction task that focuses on the local feature of the equation tree. The goal is to predict the operator between two quantity nodes in the solution tree. This is in fact a classification task with 5 potential targets, i.e., $+, -, \times, \div$ and $\wedge$. The loss function of this task is:

$$L_{OPred} = \sum_{i,j} CE(FFN([Z_i; Z_j]), op(W_i, W_j)), \tag{7}$$

where i, j are two indexes that satisfy $W_i, W_j \in W_q$ and $[Z_i; Z_j] \in \mathbb{R}^{2h}$ is the concatenation of Z_i and Z_j for the quantity W_i and W_j. $op(W_i, W_j)$ returns the operator between W_i and W_j.

Tree Distance Prediction. Another pre-training objective is to incorporate the global structure of the equation tree in a quantitative way. Inspired by Hewitt and Manning (2019), we consider the depth of each number and operator on the corresponding binary equation tree to be the key structure priori. Thus, we design another fully-supervised objective to utilize this information. More exactly, given the representation of two number nodes in an equation tree, this is a regression problem that predicts the distance (difference of their depth) between them. The loss is formulated as:

$$L_{TPred} = \sum_{i,j} MSE(FFN([Z_i; Z_j]), d(W_i, W_j)), \tag{8}$$

where $d(W_i, W_j)$ is the distance between quantity W_i and W_j in the solution tree.

The final pre-training objective is the summation of Equation 2~8 and the masked language model.

3.6 Fine-Tuning

To investigate the mathematical understanding ability of our pre-training MWP-BERT, we evaluate our model of MWP solving, quantity tagging and 7 probing tasks. Moreover, we not only use BERT but also RoBERTa (Liu et al., 2019b) as the backbone of our encoder to show the adaptiveness of proposed method.

4 Experiments

We present several empirical results with octopus evaluation settings (Bender and Koller, 2020) to prove the superiority of MWP-BERT and MWP-RoBERTa solver. In section 4.1, we illustrate the application of both solvers in the generation scenario, MWP solving, by fine-tuning them with a specific decoder (Xie and Sun, 2019). Next, we present MWP probing tasks in section 4.3 to evaluate the capability of MWP-BERT and MWP-RoBERTa on "understanding" or "capturing the meanings" of MWPs. Finally, results and analysis about ablation study are illustrated in section 4.4.

Implementation Details. We pre-train our model on 4 NVIDIA TESLA V100 graphic cards and fine-tune on 1 card. The model was pre-trained for 50 epochs (2 days) and fine-tuned for 80 epochs (1 day) with a batch size of 32. Adam optimizer (Kingma and Ba, 2014) is applied with an initial learning rate of 5e-5, which would be halved every 30 epochs. Dropout rate of 0.5 is set during training to prevent over-fitting. During testing, we use 5-beam search to get reasonable solutions. The hyper-parameters setting of our BERT and RoBERTa is 12 layers of depth, 12 heads of attention and 768 dimensions of hidden features. For the Chinese pre-training model, we use an upgrade patch of Chinese BERT and RoBERTa which are pre-trained with the whole word masking (WWM)[1] (Cui et al., 2020). For the English pre-training models, we use the official source on this website[2]. Our code and data have been open-sourced on Github [3].

4.1 MWP Solving

Experiment Settings and Datasets. Given a textual description of a mathematical problem, which contains several known variables, MWP solving targets at getting the correct answer for the corresponding question. A solver is expected to be able to predict an equation that can exactly reach the answer value. We conduct experiment based on these benchmarks, Math23k (Wang et al., 2017), MathQA (Amini et al., 2019) and Ape-210k (Zhao et al., 2020). Since there exist many noisy examples in Ape-210k, e.g., examples without equation annotations or answer values, we re-organize Ape-210k to Ape-clean and Ape-unsolvable, where the training set of Ape-clean and the whole Ape-unsolvable are used for pre-training. For the English MWP, we use the training set of MathQA (Amini et al., 2019) to perform pre-training. For the implementation of our solver, MWP-BERT is adapted as an encoder to generate intermediate MWP representation for the tree-based decoder in Xie and Sun (2019).

4.2 The Ape-clean Dataset

Ape210k is a recently released large MWPs dataset, including 210,488 problems. The problems in Ape210k are more diverse and difficult than those in Math23k as shown in 1. Not only the stronger requirement of common-sense knowledge for getting solutions, but also the missing of ground-truth solution equations or answers, will take extra obstacles for MWP solving. Among all these cases,

Unsolvable problem 1:	The price of a ball is 6 yuan, and the price of a basketball is less than 13 times of the ball's price. How much might the price of the basketball be?
Answer:	?
Unsolvable problem 2:	x is a single digit and the quotient of x72/47 is also a single digit, what is x at most?
Answer:	3
Unsolvable problem 3:	In the yard there were 25 chickens and rabbits. Together they had 80 legs. How many rabbits were in the yard?
Answer:	(80-25)*2/(4-2) = 15

Table 1: This table shows three kinds of discarded MWPs in Ape210k. The first one does not have a certain answer, and the solution of the second one cannot be represented by equations. Solving the third problem requires external constants. Thus we filter those problems out in our Ape-clean dataset.

the problems without answers can not be used for fully-supervised setting. Besides, the problems without annotated equations but only answer values can be used in the weakly-supervised learning setting. Therefore, we follow the rules below to select the usable problems from Ape210k to construct an Ape-clean dataset, which can be used for the fully-supervised learning setting. (i). We remove all MWPs that have no answer values nor equations. (ii). We remove all MWPs that only have answer values without equations. (iii). We remove all MWPs with a problem length $m > 100$ or an answer equation length $n > 20$, as they will bring obstacles for training. (iv). We remove all MWPs requiring external constants except 1 and π. (v). We remove all duplicated problems with the MWPs in Math23k, because almost all problems in Math23k can be found in Ape-210k. After data filtering, the *Ape-clean* dataset contains 81,225 MWPs, including 79,388 training problems and 1,837 testing problems. The remaining 129,263 problems in Ape210k are regarded as *Ape-unsolvable*, which can be used in the pre-training tasks in the settings of self-supervised and weakly-supervised learning.

Model Comparison. We first compare our approach with the most recent representative baselines on the benchmark Math23k dataset. The first baseline is *DNS* which is the pioneering work us-

[1]https://github.com/ymcui/Chinese-BERT-wwm

[2]https://huggingface.co/bert-base-uncased and https://huggingface.co/roberta-base

[3]https://github.com/LZhenwen/MWP-BERT

	Math23k	Math23k*	MathQA
DNS	–	58.1	–
Math-EN	66.7	–	–
GTS	75.6	74.3	71.3
NS-Solver	75.7	–	–
Graph2Tree	77.4	75.5	72.0
TSN-MD	77.4	75.1	–
KA-S2T	76.3	–	–
NumS2T	78.1	–	–
EEH-G2T	78.5	–	–
RPKHS	83.9	82.2	–
Encoder pre-train			
RoBERTa	83.5	81.7	75.3
BERT	83.8	82.0	75.1
MWP-RoBERTa	84.5	82.0	**76.6**
MWP-BERT	**84.7**	**82.4**	76.2
Seq2Seq pre-train			
REAL	82.3	80.0	–
BERT-CL	83.2	–	76.3
Gen&Rank	85.4	84.3	–

Table 2: Comparison of answer accuracy (%) among our proposed models and different baselines. Math23k column shows the results on the public test set and Math23k* is 5-fold cross validation on Math23k dataset. MathQA is adapated from Li et al. (2021); Tan et al. (2021). "RoBERTa" and "BERT" represent results without pre-training. "MWP-RoBERTa" and "MWP-BERT" represent first pre-training with proposed tasks and then fine-tuning.

ing the Seq2Seq model to solve MWPs. *Math-EN* (Wang et al., 2018) proposes an equation-normalization method and uses vanilla Seq2Seq model to get solutions. *GTS* (Xie and Sun, 2019) proposes a goal-driven tree-based decoder and achieves great results. *Graph2Tree* (Zhang et al., 2020b) constructs two graphs during data preprocessing to extract extra relationships from text descriptions. *KA-S2T* (Wu et al., 2020) proposes a novel knowledge-aware model that can incorporate background knowledge. *NS-Solver* (Qin et al., 2021) designs several auxiliary tasks to help training. *NumS2T* (Wu et al., 2021b) uses explicit numerical values instead of symbol placeholder to encode quantities. *RPKHS* (Yu et al., 2021) builds hierarchical reasoning encoder in parallel with PLM encoder. *REAL* (Huang et al., 2021) proposes a human-like analogical auxiliary learning strategy. *EEH-G2T* (Wu et al., 2021a) injects edge label information and the long-range word relationship into graph network. *Gen&Rank* (Shen et al., 2021) designs a multi-task learning framework for adapting BART in MWP solving. *BERT-CL* (Li et al., 2021) incorporates contrastive learning strategy with PLM. To avoid the implementation error that may cause unreproducible results of baseline models, we reported the results of these baselines from the papers where they were published, as many previous papers (Zhang et al., 2020b; Shen and Jin, 2020) did.

As shown in Table 2, our MWP-BERT achieves competitive results. It is worth noting that we perform strict pre-training paradigm in MWP solving, i.e., our results come from pre-training on the different annotated MWP examples that will be applied in further fine-tuning. Here, our pre-training only uses Ape-clean and Ape-unsolvable and fine-tuning only uses Math23k/MathQA.

RPKHS, *REAL* and *BERT-CL* all incorporate BERT in their model architecture, which are orthogonal to our work. Our MWP-BERT can be utilized as an MWP-specific checkpoint for their encoder part to improve their performance. Besides, *REAL*, *BERT-CL* and *Gen & Rank* are all trying to make Seq2Seq pre-training (Lewis et al., 2020), which adapt both pre-training encoder as well as pre-training decoder for MWP solving. Compared with them, our model focuses on encoder pre-training and aims at obtaining better MWP representation that can be widely applied across various MWP related tasks (like quantity tagging, MWP question generation).

Another interesting observation is that BERT-based models perform better on Chinese MWP datasets while RoBERTa-based models are good at English MWP datasets. Because the Chinese RoBERTa used in this paper is actually a BERT model that uses BERT tokenization but is trained like RoBERTa (drops the Next Sentence Prediction task). Similar behaviors can be observed in Cui et al. (2020). For English setting, RoBERTa performs better than BERT, which is consistent with conclusions raised in Liu et al. (2019b).

Evaluation on Ape-clean and Math23k when being trained by a Joint MWP Set. Moreover, we combine the training set of Math23k and Ape-clean to train MWP-BERT, and then measure the accuracy on the testing set of Math23k and Ape-clean separately. Results shown in Table 2 con-

Model	Math23k		Ape-clean	
	Equ	Ans	Equ	Ans
DNS	50.2	50.3	66.2	66.2
GTS	70.1	81.4	60.4	73.2
RoBERTa	75.8	88.8	66.7	80.2
BERT	76.7	89.4	67.0	80.4
MWP-RoBERTa	77.1	90.2	67.1	80.8
MWP-BERT	**77.5**	**91.2**	**67.5**	**81.3**

Table 3: Comparison of answer accuracy (%) between our proposed models and baselines when they are all trained by the combination of the training set from Ape-clean and Math23k dataset.

vey interesting evaluation observations. Surprisingly, the accuracy of our models on Math23k reaches above 90%, which is marvelously high (previous state-of-the-art methods can hardly reach 80% (Shen and Jin, 2020)). Compared to the results in Table 2, even *GTS* has a higher accuracy when trained with the big joint MWP set of Ape-clean and Math23k.

By comparing the performance of corresponding groups between Table 2 and Table 3, we can learn that our proposed MWP-BERT pre-training paradigm can achieve more significant boosting with more training examples, which proves the effectiveness of our proposed representation learning techniques.

4.3 Other MWP Understanding Tasks

Standard MWP solving is an equation generation task. To make a sufficient validation of the effectiveness of our model on number representations learning, MWP-specific understanding tasks are further considered. Following Hewitt and Manning (2019); Wallace et al. (2019), we design several number probing tasks and incorporate quantity tagging (Zou and Lu, 2019b) to enlarge the MWP understanding evaluation task.

Following the motivation mentioned in section 3.2, we re-run all the pre-training tasks as probing tasks to evaluate our modeling's understanding ability and test MWP-BERT in a zero-shot scenario, i.e. without fine-tuning the parameters of MWP-BERT and MWP-RoBERTa for the sake of fair comparison. We perform the probing evaluation on both Ape-clean and Ape-unsolvable, except that "OPred" and "TPred" are only evaluated on Ape-clean because they require equation solutions as the ground truth.

Model	Accuracy
QT(S)	87.3
QT(R)	88.7
QT(fix)	87.7
QT	90.8
BERT	84.5
RoBERTa	84.6
MWP-BERT	91.0
MWP-RoBERTa	**91.5**

Table 5: Comparison of tagging accuracy (%) between our proposed models and baselines.

Table 4 shows the performances of 4 different PLMs on the above mentioned MWP-specific understanding tasks. Significant improvements can be observed in all the tasks, and demonstrate the effectiveness of our proposed pre-training techniques in improving number representation of PLMs.

Besides, we borrow an MWP-specific sequence labeling task, quantity tagging (Zou and Lu, 2019b) ("QT"), to further compose MWP understanding evaluation settings. Quantity tagging (Zou and Lu, 2019a) is firstly proposed to solve MWP examples with only addition and subtraction operators in their solutions. Briefly speaking, this task requires the model to assign "+", "-" or "None" for every quantity in the problem description and can serve as an MWP understanding evaluation tool to examine the model's understanding of each variable's logic role in the reasoning flow. More exactly, this is also a classification task with 3 possible targets. We extract the corresponding vectors of all quantities according to their positions in encoded problem Z from Equation 1. Next, a 2-layer feed-forward block is connected to output the final prediction.

Following the setting in baseline method *QT* (Zou and Lu, 2019a), we perform 3-fold cross-validation and the results are given in Table 5, which shows that PLMs benefits from the proposed mathematical pre-training and outperforms the baselines.

4.4 Ablation Study

We run ablation study over the proposed training objectives to investigate the necessity for each of them. As Table 6 shows, all the proposed objectives can achieve improvements individually. Moreover, only using MLM results in weaker MWP solvers on Math23k (1.4% less) and Ape-clean (1.2% less),

	NumCount	NTGround	ATPred	CATComp	NumMComp	OPred	TPred	QT
Metric	MSE ↓	Acc ↑	Acc ↑	Acc ↑	Acc ↑	Acc ↑	MSE ↓	Acc ↑
BERT	3.08	0.87	0.75	0.77	0.77	0.50	0.97	84.5
RoBERTa	3.20	0.86	0.76	0.78	0.77	0.51	0.99	84.6
MWP-RoBERTa	0.69	0.92	0.86	0.87	0.86	0.86	0.44	91.0
MWP-BERT	0.67	0.92	0.85	0.87	0.86	0.87	0.45	91.5

Table 4: The evaluation results on MWP-specific understanding tasks. All tasks correspond to the tasks mentioned in section 4. Note that the metric for 2 tasks is mean-squared-error, while others use classification accuracy. "QT" stands for quantity tagging.

	Math23k	Ape-clean
Only MLM	89.8	80.1
Only self-supervised	90.4	80.9
w/o MLM	90.1	80.6
w/o $NumCount$	89.9	80.5
w/o $NTGround$	90.1	80.4
Only weakly-supervised	90.1	80.8
w/o $ATPred$	89.7	80.2
w/o $CATComp$	89.7	80.4
w/o $NumMComp$	89.6	80.5
Only fully-supervised	91.0	80.5
w/o $OPred$	90.5	80.3
w/o $TPred$	90.6	80.5
MWP-BERT	**91.2**	**81.3**

Table 6: The experimental results show the effectiveness of every pre-trained task. "Only self-supervised" means we only apply 3 tasks of self-supervised pre-training on the BERT encoder. We also investigate the influence of each task. For example, "w/o MLM" means only performing self-supervised pre-training and discarding the MLM pre-training task.

which again proves the effectiveness of our proposed pre-training tasks. Since the difficulty level of MWPs is usually in proportion to their solution length, we can easily identify that a set of MWPs exhibit a long-tail distribution over solution length, as well as the difficulty level, as shown in. Fig 3 of the Appendix. Thus, the 87% accuracy of human-level performance in Math23k (Wang et al., 2019) indicates that 13% of the MWPs are difficult to solve. Any solvers that can improve the accuracy above 87% are making significant contribution on solving the extremely difficult cases, such as MWPs whose solutions contain ≥ 4 variables or single variable being used multiple times. As neural models are known to be limited at dealing with these combinational and symbolic reasoning cases (Lee et al., 2020), we exam our model on these specially difficult cases. Due to the space limit, we attach several examples of these difficult cases, statistics about solution length distribution and performance for increasing length of solution equations in the Appendix. Besides, it is worth noting that even without MLM objective, our model is able to promote the PLM competitor. Besides, we can observe that linking equation structure and number during pre-training is certainly beneficial for solving MWPs.

5 Conclusion

We propose MWP-BERT, an MWP-specific PLM model with 8 pre-training objectives to solve the number representation issue in MWP. Also, a new dataset Ape-clean is curated by filtering out unsolvable problems from Ape210k, and the filtered MWPs are useful for self- and weakly-supervised pre-training. Experimental results show the superiority of our proposed MWP-BERT across various downstream tasks on generation and understanding. In terms of the most representative task MWP solving, our approach achieves the highest accuracy, and firstly beats human performance. Better numerical understanding ability is also demonstrated in the probing evaluation. We believe that our study can serve as a useful pre-trained pipeline and a strong baseline in the MWP community.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *NAACL*, pages 2357–2367.

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. In *EMNLP*, pages 5946–5951.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: on meaning, form, and understanding in the age of data. In *ACL*, pages 5185–5198.

Ting-Rui Chiang and Yun-Nung Chen. 2018. Semantically-aligned equation generation for solving and reasoning math word problems. In *NAACL*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *EMNLP: Findings*, pages 657–668.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*, pages 2368–2378.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *ACL*, pages 946–958.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *NAACL*, pages 4129–4138.

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

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

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *ACL*, pages 887–896.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. In *Findings of EMNLP*, pages 786–796.

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

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

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

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

Dennis Lee, Christian Szegedy, Markus N. Rabe, Sarah M. Loos, and Kshitij Bansal. 2020. Mathematical reasoning in latent space. In *ICLR*. OpenReview.net.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

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

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Zhou, Chao Li, Hongzhi Liu, and Yunbo Cao. 2021. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. *arXiv preprint arXiv:2110.08464*.

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

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

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

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. In *ACL*, pages 5870–5881.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *EMNLP*, pages 3780–3789.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *EMNLP*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *COLING*, pages 2924–2934.

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

Minghuan Tan, Lei Wang, Lingxiao Jiang, and Jing Jiang. 2021. Investigating math word problems using pretrained multilingual language models. *CoRR*, abs/2105.08928.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro A. Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *NAACL-HLT*, pages 644–656.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *EMNLP*, pages 5306–5314.

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

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

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

Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuan-Jing Huang. 2020. A knowledge-aware sequence-to-tree network for math word problem solving. In *EMNLP*, pages 7137–7146.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021a. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of EMNLP*, pages 1473–1482.

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2021b. Math word problem solving with explicit numerical values. In *ACL*, pages 5859–5869.

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

Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *EMNLP*, pages 3384–3394.

Ma Yuhui, Zhou Ying, Cui Guangzuo, Ren Yun, and Huang Ronghuai. 2010. Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. In *2010 Second International Workshop on Education Technology and Computer Science*, volume 2, pages 476–479. IEEE.

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

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. Graph-to-tree learning for solving math word problems. In *ACL*, pages 3928–3937.

Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020c. Do language embeddings capture scales? In *Findings of EMNLP*, volume EMNLP 2020, pages 4889–4896. Association for Computational Linguistics.

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

Yanyan Zou and Wei Lu. 2019a. Quantity tagger: A latent-variable sequence labeling approach to solving addition-subtraction word problems. In *ACL*, pages 5246–5251.

Yanyan Zou and Wei Lu. 2019b. Text2math: End-to-end parsing text into math expressions. In *EMNLP*, pages 5330–5340.

Problem 1:	There are 20 questions in an exam. Solving a question correctly gets 5 points, and 1 point is deducted if the answer is wrong. Jack gets 70 points. How many questions did he get right?
Answer:	20-(20*5-70)/(5+1)
Problem 2:	Peter is reading a book. He reads 30% of the whole book on the first day, and 15 pages on the second day. The ratio of the number of pages that has been read to the number of pages not read is 2:3. How many pages does this book have?
Answer:	15/(1-((3)/(2+3))-30%)
Problem 3:	There are 72% of 50 students can swim, and (3/5) of 25 girls can swim, how many percent of the boys can swim?
Answer:	(50*72%-25*(3/5))/(50-25)

Table 7: This table shows three difficult problems in Math23k.

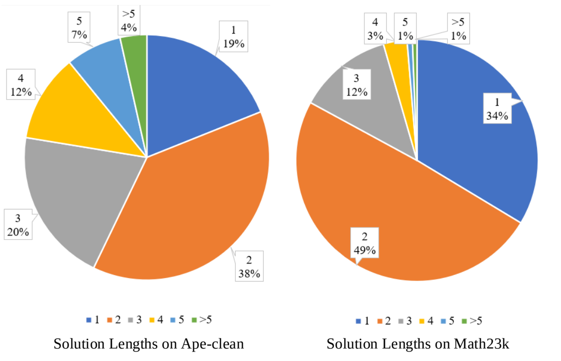


Figure 3: The solution length distributions on Math23k and Ape-clean.

#op	#P	BERT	MWP-BERT
0	16	100	100
1	331	95.1	96.3
2	485	90.7	90.7
3	124	79.8	82.3
4	31	58.0	67.7
5	7	85.7	85.7
>5	6	33.3	50

Table 8: The answer accuracy of BERT and MWP-BERT on problems with different lengths in Math23k. #op denotes the number of operators in the solution. #P is the number of problems of that kind of MWPs in the public test set of Math23k.

A Appendix

A.1 Accuracy w.r.t. Solution Length

To better understand the improvement of the MWP solving performance of our model, we evaluate the problems with different lengths of solutions separately. The solution distribution details can be found in Figure 3 It is expected that getting longer solutions requires more comprehensive understanding and complex reasoning, like the three difficult examples shown in Table 7. The results in Table 8 demonstrate that our proposed MWP-BERT overcomes more difficult problems than the vanilla BERT model. Although the statistical improvement from BERT to MWP-BERT is marginal, our method really enhances the mathematical understanding and reasoning ability of PLMs.

A.2 Case Study

We perform case study as shown in Table 9. Firstly, we choose a difficult problem from Math23k dataset and use 3 different solvers to solve it. Both GTS (Xie and Sun, 2019) and Graph2Tree (Zhang et al., 2020b) fail to generate the right solution for it, while our proposed MWP-BERT solves it correctly. This example shows that our encoder has a stronger capability to understand complex MWPs to guide the tree-based decoder generate correct solutions. Secondly, when solving a pair of 2 similar problems (i.e., problem 1 and problem 2 in Table 9), GTS, Graph2Tree and our MWP-BERT successfully solve the former problem. However, the baseline methods GTS and Graph2Tree both fail to solve the latter one. Our MWP-BERT generates the correct answer. This example proves that our probing tasks help the encoder to capture minor variations inside the problem description, leading to more accurate solutions.

Difficult Problem:	There are totally 48 cars and motorcycles in a parking lot. Each car has 4 wheels and each motorcycle has 3 wheels. If they have 172 wheels in total. How many motorcycles are there in the parking lot?
GTS:	$x = 48 + (172 - 48)/(4 - 3)$ (✗)
Graph2Tree :	$x = 48 - (48 - 172)/3$ (✗)
MWP-BERT:	$x = (48 * 4 - 172)/(4 - 3)$ (✓)
Problem 1:	Team A and team B are working on a project together. Team A finished (4/15) of the project, and team B finished (2/15) more than Team A . How many percentage did the two teams finish in total?
GTS:	$x = (4/15) + (2/15) + (4/15)$ (✓)
Graph2Tree :	$x = (4/15) + (2/15) + (4/15)$ (✓)
MWP-BERT:	$x = (4/15) + (2/15) + (4/15)$ (✓)
Problem 2:	Team A and team B are building a road. Team A builds (4/9), and team B builds (1/9) more than team A. How many percentage does Team B build?
GTS:	$x = (4/9) + (1/9) + (4/9)$ (✗)
Graph2Tree :	$x = (4/9) + (1/9) + (4/9)$ (✗)
MWP-BERT:	$x = (4/9) + (1/9)$ (✓)

Table 9: Our case study.