# Unbiased Math Word Problems Benchmark for Mitigating Solving Bias

**Zhicheng Yang**[1,2], **Jinghui Qin**[2], **Jiaqi Chen**[2,3] and **Xiaodan Liang**[1,2]*
[1]Shenzhen Campus of Sun Yat-sen University
[2]Sun Yat-sen University, [3]Dark Matter AI Inc.
{yangzhch6,qinjingh}@mail2.sysu.edu.cn
{jadgechen,xdliang328}@gmail.com

## Abstract

In this paper, we revisit the solving bias when evaluating models on current Math Word Problem (MWP) benchmarks. However, current solvers exist solving bias which consists of data bias and learning bias due to biased dataset and improper training strategy. Our experiments verify MWP solvers are easy to be biased by the biased training datasets which do not cover diverse questions for each problem narrative of all MWPs, thus a solver can only learn shallow heuristics rather than deep semantics for understanding problems. Besides, an MWP can be naturally solved by multiple equivalent equations while current datasets take only one of the equivalent equations as ground truth, forcing the model to match the labeled ground truth and ignoring other equivalent equations. Here, we first introduce a novel MWP dataset named UnbiasedMWP which is constructed by varying the grounded expressions in our collected data and annotating them with corresponding multiple new questions manually. Then, to further mitigate learning bias, we propose a Dynamic Target Selection (DTS) Strategy to dynamically select more suitable target expressions according to the longest prefix match between the current model output and candidate equivalent equations which are obtained by applying commutative law during training. The results show that our UnbiasedMWP has significantly fewer biases than its original data and other datasets, posing a promising benchmark for fairly evaluating the solvers' reasoning skills rather than matching nearest neighbors. And the solvers trained with our DTS achieve higher accuracies on multiple MWP benchmarks. The source code is available at https://github.com/yangzhch6/UnbiasedMWP.

## 1 Introduction

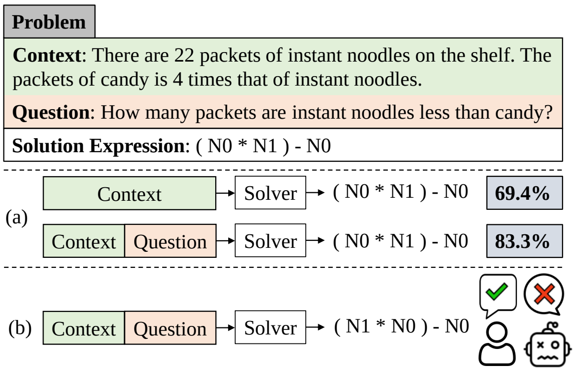

Problem
Context: There are 22 packets of instant noodles on the shelf. The packets of candy is 4 times that of instant noodles.
Question: How many packets are instant noodles less than candy?
Solution Expression: ( N0 * N1 ) - N0
(a)
Context
Solver
( N0 * N1 ) - N0
69.4%
Context
Question
Solver
( N0 * N1 ) - N0
83.3%
(b)
Context
Question
Solver
( N1 * N0 ) - N0

Figure 1: Illustration of solving bias in MWP. A typical MWP problem can be divided into context and question. (a) shows that 69.4% of the problems in Math23K can be answered by the solver (Bert2Tree) without looking at the question, verifying its severe data bias. (b) shows that the current training procedure ignores the equivalent expressions, indicating the possible learning bias.

Math Word Problem (MWP) solving is a long-standing challenging task in Natural Language Processing (NLP) and has attracted lots of attention recently (Upadhyay and Chang, 2017; Upadhyay et al., 2016; Huang et al., 2018; Wang et al., 2017, 2018, 2019; Qin et al., 2020; Huang et al., 2021; Shen et al., 2021; Qin et al., 2021; Wu et al., 2021). An automatic MWP solver should not only understand the problem's semantic information but also reason the grounded mathematical relationships implicit in the problem, so that it can transform natural language into solution expression.

More recently, deep learning methods (Wang et al., 2017, 2018; Huang et al., 2021; Shen et al., 2021; Wu et al., 2021) have made great progress in MWP solving and achieved impressive results on several popular benchmarks, such as Math23K (Wang et al., 2017) and MAWPS (Koncel-Kedziorski et al., 2016). However, there exists some severe possible solving bias in these benchmarks, consisting of data bias and learning bias. Here, the **data bias** is introduced since the training dataset does not fully cover diverse questions for each problem narrative of all MWPs, leading to the situation that a solver only learns shallow heuristics rather than deep seman-

[*] Corresponding author.

*Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1401 - 1408
July 10-15, 2022 ©2022 Association for Computational Linguistics

tics for understanding problems. Besides, even the question of an MWP is deleted, a solver still can solve it correctly, as shown in Figure 1(a). On the other hand, an MWP can be solved by multiple equivalent equations while current popular datasets only take one of the equivalent equations as the ground truth output for each sample, forcing the model to learn the labeled ground truth and ignore other equivalent equations which may be more suitable for a solver to learn, leading to **learning bias** during training. As shown in Figure 1(b), if a solver may generate an answer-corrected expression that is different from ground-truth expression, it will be thought an error and the loss between the answer-corrected expression and the ground-truth expression will be back-propagated to the solver during training, leading to over-correct the solver. This learning bias makes it harder to learn to reason out answer-corrected expressions.

To mitigate the solver bias for pushing advanced models to learn underlying reasoning skills rather than solely matching nearest results, we first build a novel MWP dataset, UnbiasedMWP, to cover diverse questions for each problem narrative of all MWPs. It is constructed by varying the grounded expressions in our collected data and annotating them with corresponding new questions manually, thus mitigating data bias. Then, to mitigate the learning bias, we propose a Dynamic Target Selection (DTS) Strategy to dynamically select the most suitable target expression by applying the longest prefix match between the current model output and candidate equivalent equations obtained by applying commutative law during training. Our experimental result shows that our UnbiasedMWP has significantly fewer biases than its original data and other datasets, and the solvers equipped with our equivalent expression matching loss can achieve higher accuracy on multiple MWP benchmarks such as Math23K and our UnbiasedMWP. Our main contributions are in two folds:

- We propose a large-scale data-unbiased dataset named UnbiasedMWP consisting of 10264 MWPs with diverse questions. The dataset is constructed by varying the grounded expressions and annotated corresponding questions. With this dataset, we can force a model to learn deep semantics rather than shallow heuristics for solving an MWP.
- We propose a Dynamic Target Selection (DTS) Strategy to dynamically select a more suitable target expression, thus eliminating the learning bias caused by ignoring equivalent expressions during the training procedure. Experimental results demonstrate that the models trained with DTS achieve better performances on multiple benchmarks. Our DTS can improve the baseline model up to 1%, 2.5%, and 1.5% on Math23K, UnbiasedMWP-Source, and UnbiasedMWP-All, respectively.

## 2 UnbiasedMWP dataset

In this section, we introduce the construction procedure of our UnbiasedMWP dataset. Based on the newly-collected raw data, we design a pipeline for pre-processing and rewriting questions according to formula variations, which is strictly performed by the annotators to obtain unbiased data.

### 2.1 Data Collection and Pre-processing

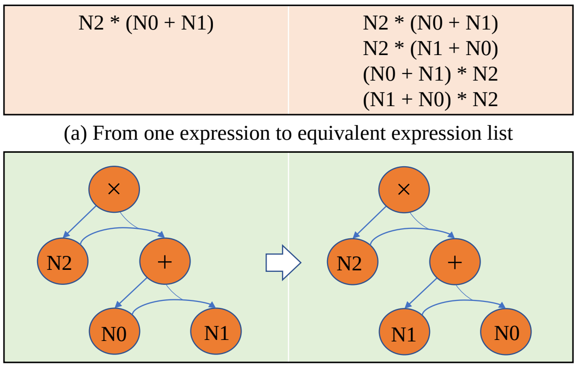
To collect UnbiasedMWP, we crawl 2907 examples from an online education website[1]. During pre-processing, the number mapping (Wang et al., 2017) is deployed to replace the numbers in solution expression with symbolic variables (e.g., $N0$, $N1$). Then, the workers are asked to split the problem text into two parts: context (a narrative implicated with numerical relationships) and question (a short text that requires the solution of a mathematical relationship).

### 2.2 Expression Variation

As shown in Figure 1, a neural network model can solve problems even without questions, this shows that a solver solves problems mainly by relying on shallow heuristics rather than deep semantic understanding. Besides, current popular and large-scale datasets do not fully cover any possible questions for the context in each MWP, which also results in data bias. To mitigate this issue, we annotate each narrative with various possible questions to construct an unbiased MWP benchmark by enumerating various expressions according to the number in the context, asking workers to design questions for each expression. If an expression can not be assigned with a suitable question, we remove it.

To enumerate various possible expressions, we design three types of variation to create different expressions for each context: **Variable assortment (Va) variations**: Selecting two variables from the context and combining them with the operators

[1] https://damolx.com/



(a) From one expression to equivalent expression list

(b) One of the example generation procedure

Figure 2: Equivalent Expression Tree Generation. (a) shows the results of generation, (b) shows one of the generation examples.

"$+, -, *, /$", such as $n_0 + n_1$, $n_0 - n_1$, etc. **Sub-expression (Sub) variations**: From the original expression, we choose all sub-expressions of it and change the operators to get new expressions. **Whole-expression (Whole) variations**: We get new expressions by changing the operators in the original expression. Besides, workers also can propose new expressions and annotate them.

Various expressions are first acquired by applying the variation processing. Then, we ask workers to write a practical question for each meaningful expression variation. For those meaningless expressions that can not be annotated with any practical question, we filtered out them. The details of data split and statistics are listed in the appendix.

## 3 Dynamic Target Selection Strategy

During the common MWP training procedure, only one expression is used as ground truth while the equivalent expressions are ignored. Consider the following case: the ground truth label is "$(N1 * N0) - N0$" while the model output is "$(N0 * N1) - N0$". Although they are mathematical equivalent, the model output is judged to be incorrect. Therefore, models are prone to be biased during training. To address this issue, we generate the equivalent expressions of the original ground truth expression and then select an equivalent expression matching the longest prefix with the current model output as target expression in the training procedure.

### 3.1 Equivalent Expression Tree Generation

To generate equivalent expressions, we consider swapping sub-expressions on the two sides of symmetric binary operators such as: $+$ and $\times$. Firstly, we construct an expression tree for each expression following (Xie and Sun, 2019). Then, we recursively examine each operator node from bottom to up and swap the left and right sub-trees of the node if it is a symmetric operator, and then we add the result new tree to a list. Finally, we iterate all the trees in the list into infix or prefix expressions to get multiple equivalent expressions. The generation procedure is illustrated in Algorithm 1. An example of the generation is illustrated in Figure 2 (b), we exchange the position of 'N0' and 'N1', and get a new equivalent expression. An example of generated equivalent expressions is shown in Fig. 2 (a).

**Algorithm 1:** Equivalent Expression Tree Generation

```
Function: Variation(tree, root, equList)
Input: Expression tree: $tree$; Root node of the input
       tree: $root$;
Output: Equivalent expression list: equList.
if $root$ is null then
    return
Variation(tree, root.left, equList)
Variation(tree, root.right, equList)
if root.value is symmetric operator then
    swap(root.left, root.right)
    equList.append(tree)
    Variation(tree, root.left, equList)
    Variation(tree, root.right, equList)
    swap(root.left, root.right)
return
```

### 3.2 Dynamic Target Selection (DTS)

During the training procedure, the solver may generate the correct start part expression which matches the prefix of one of the equivalent expressions but not matches the prefix of the ground truth labeled in the dataset. If we still use the ground truth as the target to train the solver, this will lead to oversize error to correct the model prediction, leading to sub-optimal learning and learning bias. To mitigate this issue, we dynamically choose a new equivalent target expression as a training target that can match the current model output with the longest prefix. In this way, the loss will not be oversized so that we can make the solver easier to solve problems correctly.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets.** We conduct experiments on Math23K (Wang et al., 2017) dataset and our UnbiasedMWP dataset. We use **UnbiasedMWP-Source** to represent the initial collection of samples while using

| | Math23K | | UnbiasedMWP -Source | | UnbiasedMWP -All | |
|---|---|---|---|---|---|---|
| Methods | Ori | Del_q | Ori | Del_q | Ori | Del_q |
| Math-EN | 68.4 | 55.2 | 62.0 | 42.5 | 52.5 | 20.4 |
| Group-Attn | 69.5 | 57.9 | 61.5 | 42.0 | 53.1 | 20.1 |
| GTS | 75.6 | 61.9 | 64.5 | 49.0 | 63.6 | 25.1 |
| Graph2Tree | 77.4 | 63.2 | 65.0 | 50.0 | 64.0 | 25.0 |
| Bert2Tree | 83.3 | 69.4 | 73.0 | 55.0 | 78.1 | 22.5 |

Table 1: Experimental results on Math23K, UnbiasedMWP-Source, and UnbiasedMWP-All. Ori indicates the original data and the Del_q indicates data with the question removed.

**UnbiasedMWP-All** to represent the initial collection of samples and their various variations.

**Baselines.** We validate our UnbiasedMWP dataset and DTS training strategy with multiple models: **Math-EN** (Wang et al., 2018): a seq2seq model with equation normalization for reducing target space. **GROUPATT** (Li et al., 2019): a solver borrowing the idea of multi-head attention from Transformer (Vaswani et al., 2017). **GTS** (Xie and Sun, 2019): a tree-structured neural network in a goal-driven manner to generate expression trees. **Graph2Tree** (Zhang et al., 2020): an enhanced GTS with quantity graph. **BERT2Tree**: a strong baseline we constructed by replacing RNN encoder with BERTEncoder(Cui et al., 2020) in GTS. More details can be referred to the appendix.

### 4.2 Experimental Results

**Bias Analysis on MWP Datasets** We conduct similar experiments in (Patel et al., 2021) by removing question text on Math23K datasets and our collected UnbiasedMWP source data to show the solver mainly relied on shallow heuristics. As shown in Table 1, the experimental results on Math23K and UnbiasedMWP-Source show that all models still perform well even lack the question information. This suggests the patterns in the context have a strong correlation with the output expression, thus causing the model to learn bias in MWPs. We also conduct the same experiments on the UnbiasedMWP-All dataset. From Table 1, we can observe that the accuracies of the MWP without questions (Del_q) are significantly lower on UnbiasedMWP-All than the other two datasets. This shows that our UnbiasedMWP can force the solver to solve an MWP with less bias.

**Robustness Analysis** To further validate the advantages of our different variation data and how to improve a solver's robustness, we train two solvers on UnbiasedMWP-Source (Src) and UnbiasedMWP-All (All) and compare their performances on different test sets (Src, Src+Va, Src+Sub, Src+Whole, and All). From Table 2, we can observe that the solver trained with different variation data is more robust than the solver trained only with the initially collected samples on various test sets. This shows that our UnbiasedMWP can mitigate the learning bias of an MWP solver.

| Train \ Test | Src | Src+Va | Src+Sub | Src+Whole | All |
|---|---|---|---|---|---|
| Src | 73.0 | 37.3 | 49.7 | 53.1 | 34.9 |
| All | **75.5** | **82.4** | **79.5** | **71.1** | **78.1** |

Table 2: Comparison of results using different training and testing set. *Va*, *Sub*, and *Whole* stand for the three variations mentioned in Section 2.2. *All* denotes combining all three variations (*Va* + *Sub* + *Whole*) on source (*Src*) dataset.

| Methods | DTS | Math23K | UnbiasedMWP -Source | UnbiasedMWP -All |
|---|---|---|---|---|
| GTS | ✗ | 75.6 | 64.5 | 63.6 |
| GTS | ✓ | **76.4** | **65.5** | **63.7** |
| Graph2Tree | ✗ | 77.4 | 65.0 | 64.0 |
| Graph2Tree | ✓ | **77.8** | **65.5** | **64.6** |
| Bert2Tree | ✗ | 83.3 | 73.0 | 78.1 |
| Bert2Tree | ✓ | **84.3** | **75.5** | **79.6** |

Table 3: Comparison of experimental results with or without DTS of GTS-based (Xie and Sun, 2019) model.

**Analysis on DTS strategy** We conduct our DTS training strategy on Math23K and UnbiasedMWP. As shown in Table 3, our DTS training strategy helps several models achieve better performance. Especially, our DTS improves the accuracy of the Bert2Tree model from 83.3% to 84.3% on Math23K, from 73.0% to 75.5% on UnbiasedMWP-Source, and from 78.1% to 79.6% on UnbiasedMWP-All. In summary, the experimental results verify the validity of our DTS strategy.

## 5 Conclusion

In this paper, we revisit the solving bias in MWP. To mitigate the data bias caused by lacking question diversity, we construct a data set called UnbiasedMWP by variating the expressions in new-collected data. The experimental results illustrate that the solver trained on UnbiasedMWP is more robust than on our collected data. To mitigate the learning bias caused by loss overcorrect with taking only one ground-truth, we proposed a strategy to generate the equivalent expressions and select the longest prefix with the current model output during training, called Dynamic Target Selection (DTS). Experimental results show that our DTS helps several models achieve better performance.

## Acknowledgements

This work was supported in part by National Key R&D Program of China under Grant No. 2020AAA0109700, National Natural Science Foundation of China (NSFC) under Grant No.U19A2073 and No.61976233, Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, National Natural Science Foundation of Guangdong Province (Grant No.2022A1515011835), China Postdoctoral Science Foundation funded project (Grant No. 2021M703687), Shenzhen Fundamental Research Program (Project No. RCYX20200714114642083, No. JCYJ20190807154211365), CAAI-Huawei MindSpore Open Fund. We also thank MindSpore for the partial support of this work, which is a new deep learning computing framework[2].

## References

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for Chinese natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 657–668, Online. Association for Computational Linguistics.

Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, and Song-Chun. Zhu. 2021. Smart: A situation model for algebra story problems via attributed grammar. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223. Association for Computational Linguistics.

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 the Association for Computational Linguistics: EMNLP 2021*, pages 786–796, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial examples for evaluating math word problem solvers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2705–2712, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167, Florence, Italy. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (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 *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306. Association for Computational Linguistics.

Shyam Upadhyay and Mingwei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 494–504.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[2] https://www.mindspore.cn/

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069. Association for Computational Linguistics.

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

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854. Association for Computational Linguistics.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1473–1482, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhipeng Xie and Shichao Sun. 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-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.

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

# A Appendix

## A.1 Data Split

To ensure that the model does not see the context from the /testing set during training, We first split the training, validation, and testing set on our newly collected source dataset. Then we further apply the expression variation (mentioned in Section 2.2) to expand the data on different subsets. The size of the split of our collected data and variation data is shown in Table 4.

| Split | UnbiasedMWP<br>-Source | UnbiasedMWP<br>-All |
|---|---|---|
| Train | 2507 | 8895 |
| Validation | 200 | 684 |
| Test | 200 | 685 |

Table 4: Size of UnbiasedMWP data split.

## A.2 Examples of data variation

Figure 3 shows some examples of our data variation.

**Example 1**

**Context**: There were 892 tourists in the morning, 255 left at noon, and 304 came in the afternoon.

**Question**: How many tourists were there at this time?

**Solution Expression**: ( 892 - 255 ) + 304

**Variation**:

(1) How many times as many tourists arrive in the afternoon as leave at noon? —— 304 / 255

(2) How many more tourists came in the afternoon than left at noon? —— 304 - 255

(3) How many times as many tourists come in the morning as in the afternoon? —— 892 /=/ 304

(4) How many more tourists came in the morning than in the afternoon? —— 892 - 304

(5) How many tourists came to the science park on this day? —— 892 + 304

(6) How many tourists were left at noon? —— 892 - 255

**Example 2**

**Context**: The school has 26 basketballs. There are 4 fewer volleyballs than 12 times as many basketballs.

**Question**: How many volleyballs are there?

**Solution Expression**: ( 26 * 12 ) - 4

**Variation**:

(1) How many volleyballs and basketballs are there? —— 26 + ( ( 26 * 12 ) - 4 )

(2) How many more volleyballs are there than basketballs? —— ( ( 26 * 12 ) - 4 ) - 26

**Example 3**

**Context**: 6 groups from class A of a primary school donated $624 to the earthquake-stricken area, while 5 groups from Class B donated A total of $705 yuan.

**Question**: What is the average donation per group in Class A?

**Solution Expression**: 624 / 6

**Variation**:

(1) How much did the two classes contribute altogether? —— 624 + 705

(2) How many times did Class A donate as much as class B? —— 624 / 705

(3) How much more did Class B donate than Class A? —— 705 – 624

(4) How much does the average group in Class B donate? —— 705 / 5

(5) How much more per group did Class B donate than class A? —— ( 705 / 5 ) - ( 624 / 6 )

(6) How many times did the average group in Class B donate as much as the average group in Class A? —— ( 705 / 5 ) / ( 624 / 6 )

Figure 3: Some examples of our data variation.

## A.3 Data statistic

We analyze the proportions of data of different prefix expression lengths in UnbiasedMWP dataset and the result is shown in Table 5. We analyze our UnbiasedMWP to count the size of different variation data, the statistical result is shown in Table 6. Note that the count of All data is not equal to the sum of the above rows in the table, because there will be some overlap between the variation data obtained in the three data variation methods mentioned in Section 2.2.

| Expression Length | 3 | 5 | 7 | 9 | 11 | >11 |
|---|---|---|---|---|---|---|
| Count | 6357 | 2560 | 1011 | 215 | 90 | 31 |

Table 5: Statistics analyse on prefix expression length.

| | Count |
|---|---|
| Source | 2907 |
| Variable assortment | 5083 |
| Sub-expression | 2843 |
| Whole expression | 2205 |
| All data | 10264 |

Table 6: Statistics analyse on variation data.

We also analyze the accuracy of data of different prefix expression lengths for Bert2Tree model shown in Table 7. Experimental results show that the longer the expression, the lower the accuracy.

| Expression Length | 3 | 5 | 7 | >= 9 |
|---|---|---|---|---|
| Count | 85.3 | 74.3 | 60.30 | 26.5 |

Table 7: Performance of Bert2Tree on different prefix expression length of UnbiasedMWP-All.

## A.4 Implementation details

Pytorch[3] is used to implement our our MWP solver on Linux with NVIDIA RTX1080Ti GPU card. Our **Bert2Tree** model is constructed by replacing the encoder in **GTS** model with the Chinese Bert(Cui et al., 2020). The learning rate is set as $5e^{-5}$ and $1e^{-3}$ for Bert encoder and tree-decoder respectively. Adam is set as the optimizer of **Bert2Tree** while $\beta_1$ = 0.9, $\beta_2$ =0.999, and $\epsilon$ = $1e^{-8}$. The batch size is 32. Dropout weight is set as $0.5$ with weight decay $1e^{-5}$. For the other four models, **Math-EN**, **Group-Attn**, **GTS** and **Graph2Tree**, we follow their original parameter settings in (Hong et al., 2021). Since the data preprocessing code in **Graph2Tree** is not open, we do not evaluate this model on our own data.

In the experiments, we train Bert2Tree for 100 epochs on Math23K while 50 epochs on our UnbiasedMWP-Source and UnbiasedMWP-All data, because Math23K is a larger benchmark dataset whch contains 23K samples. For the Del_q experiments, We intercept the last sentence (question) by detecting punctuation marks in Math23K which may cause some very small errors but does not affect the overall results of the experiment. For our UnbiasedMWP dataset, we directly use the context to do the Del_q experiment.

## A.5 Related Work

### Math Word Problem Solving

In recent years, deep learning models especially Seq2Seq models(Wang et al., 2017; Li et al., 2019; Wang et al., 2018; Xie and Sun, 2019; Zhang et al., 2020; Qin et al., 2021; Shen et al., 2021; Wu et al., 2021), have made great progress in MWPs by learning to translate problem text in natural language into mathematical solution expression. (Wang et al., 2017) is the first to apply deep learning in MWPs and propose a widely used dataset called Math23K. (Li et al., 2019) propose a group attention mechanism to extract multi-dimensional features. (Xie and Sun, 2019) propose a tree decoder to decode expression as prefix order. Based on (Xie and Sun, 2019), (Zhang et al., 2020) improve the encoder embedding by fusing a graph encoder's output. (Qin et al., 2021) propose a framework by applying multiple auxiliary tasks to improve the problem embedding and the ability to predict commonsense constants. (Shen et al., 2021) devise a new ranking task for MWP and propose the Generate & Rank, a multi-task framework based on a generative pretrained language model. (Wu et al., 2021) propose a novel Edge-Enhanced Hierarchical Graph-to-Tree model (EEH-G2T), in which the math word problems are represented as edge-labeled graphs.

**Challenging Datasets and Adversarial Examples of MWP** More challenging datasets in MWP are proposed in recent years, Ape210K (Zhao et al., 2020) provides a large-scale benchmark for evaluating MWP solvers, HMWP (Qin et al., 2020) is a Chinese MWP benchmark including examples with multiple-unknown variables requiring non-linear equations to solve.

Although solvers have achieved impressive performance on these datasets, the robustness of the

[3] http://pytorch.org

solvers is questioned in (Kumar et al., 2021). Besides, (Patel et al., 2021) also points out that MWP solvers rely on shallow heuristics to achieve high performance and propose SVAMWP dataset. SVAMWP is more reliable and robust for measuring the performance of MWP solvers, because it raises the requirement for the model's sensitivity to question text through applying variations over word problems. Unlike SVAMWP, our variations are applied to expressions to get different expression-question pairs.

### A.6 Analysis on Effects of PLM

We conduct experiments on our UnbiasedMWP dataset as shown in Table 1. In the experiments, for models without PLMs such as **Math-EN** (Wang et al., 2017), **GROUPATT** (Li et al., 2019), **GTS** (Xie and Sun, 2019), **Graph2Tree** (Zhang et al., 2020), they perform worse on UnbiaseMWP-All dataset than on UnbiasedMWP-Source dataset, whereas for **Bert2Tree** model with Bert (Cui et al., 2020), it performs significantly better on UnbiasedMWP-All dataset. This shows that the UnbiasedMWP-ALL dataset with diverse questions is more likely to confuse the model, because the context of the sample remains unchanged and only changes the question. However, Bert2tree can better distinguish the difference between diverse questions through the pre-trained language model.