Improving Compositional Generalization in Math Word Problem Solving

Yunshi Lan*
East China Normal University
yslan@dase.ecnu.edu.cn

Lei Wang*
Singapore Management University
lei.wang.2019@phdcs.smu.edu.sg

Jing Jiang
Singapore Management University
jingjiang@smu.edu.sg

Ee-Peng Lim
Singapore Management University
eplim@smu.edu.sg

Abstract

Compositional generalization refers to a model's capability to generalize to newly composed input data based on the data components observed during training. It has triggered a series of compositional generalization analysis on different tasks as generalization is an important aspect of language and problem solving skills. However, the similar discussion on math word problems (MWPs) is limited. In this manuscript, we study compositional generalization in MWP solving. Specifically, we first introduce a data splitting method to create compositional splits from existing MWP datasets. Meanwhile, we synthesize data to isolate the effect of compositions. To improve the compositional generalization in MWP solving, we propose an iterative data augmentation method that includes diverse compositional variation into training data and could collaborate with MWP methods. During the evaluation, we examine a set of methods and find all of them encounter severe performance loss on the evaluated datasets. We also find our data augmentation method could significantly improve the compositional generalization of general MWP methods. Code is available at https://github.com/demoleiwang/CGMWP.

1 Introduction

Math Word Problem (MWP) solving can be formulated as a generation task whose goal is to generate an abstract expression to solve a given MWP. For example, to solve MWP (a) in Figure 1, we want to be able to generate the correct expression "$12 + 3$". Among the methods proposed to solve MWPs [20, 31, 37, 39], many leveraged advanced neural networks and have achieved promising results. Like many other tasks such as Question Answering [33, 15], MWP solving methods are expected to exhibit *Compositional Generalization*, an important capability to handle novel compositions of known components after learning the "rules of composition" from the training data. Prior work has shown that models often fail to capture the underlying compositional structure and suffer from big loss of the performance on the data splits with a large compositional gap [8, 14].

Without exception, an ideal MWP model is expected to correctly infer novel compositions of seen component expressions to answer more new math problems in the text. Assume *Constituents*, which are the concepts described as background scenario in MWPs (e.g. "apples", "number", "price"), are seen in both training and test sets. If MWP (a) and (b) in Figure 1 are contained in training set, we might encounter following two types of novel *Compositions* during prediction: (1) Test set contains more complex combinations of expression templates that are not observed. For example, constituents

*Yunshi Lan and Lei Wang contribute equally to this work.

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on MATH-AI.

Train	(a)	There are 12 apples, the number of peaches is 3. What's total number of apples and peaches? $12 + 3$
	(b)	The price of one notebook is \$36, the price of one pencil is \$2. What's the ratio of prices of one notebook and one pencil? $36/2$
Test	(c)	There are 12 apples, the number of peaches is 3, the number of notebook is 2. What is the total number of apples, peaches, and notebooks? $12 + 3 + 2$
	(d)	There are 3 apples, the number of peaches is 2. What's the ratio of apples and peaches? $3/2$

Figure 1: MWP examples with compositional challenges.

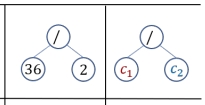
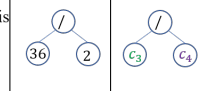
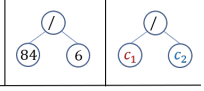
(*a*) There **are 36 apples**, the number of **peaches is 2**. What is the ratio of apples and peaches? Expression: 36 / 2	
(*b*) The price of **one notebook is \$36**, the price of **one pencil is \$2**. What is the ratio of prices of one notebook and one pencil? Expression: 36 / 2	
(*c*) These **are 84 apples**, the number of **peaches is 6**. What is the ratio of apples and peaches? Expression: 84 /6	

Figure 2: Three graph construction examples for DBCA.

in (c) have occurred in the training set, but (c) queries about an unseen expression template, which is a compound version of (a). (2) Test set contains unseen combinations of expression templates and constituents. In Figure 1, (d) describes the same scenario as (a) but queries about an expression template that has not occurred in this scenario but occurred in a different scenario like (b) during training. In the perspective of semantic interpretation, these two types of compositions require structural and lexical generalization of models [16], respectively. There are only few researches research works that studied the out-of-distribution (OOD) phenomenon issues on MWPs [11, 19, 28], their discussion is limited to length divergence or text perturbation of data, which is only tangentially related to compositional generalization.

This manuscript aims to study compositional generation on MWP solving. Specifically, we investigate the problem with the following steps: 1) we setup the problems by creating compositional challenges on MWPs. First, we adopt a data splitting method based on DBCA [33] to maximize the compositional gap between training and test for Math23K [38] and MAWPS [18] datasets. Then we synthesize data via rules to isolate the effect of each type of novel compositions, which results in a synthetic data set with a clear partition of compositions; 2) we propose an effective data augmentation method that could be collaborated with any MWP methods to improve their compositional generalization capabilities. Specifically, the data augmentation procedure is conducted in an iterative manner, where a well-trained MWP solver is leveraged to select the high-qualified augmented data and involve them into MWP corpus for the next round of data augmentation; 3) we examine a set of existing methods as well as our data augmentation method and analyze how the mechanisms make them robust or fragile to the compositional challenges.

2 Problem Setup

The goal of MWP solving is to generate an expression e based on q, which is formed by quantities and pre-defined mathematical operators and which, when evaluated, gives the numerical answer to q. We are supposed to leverage $\mathcal{D}^p = \{(q_1^p, e_1^p), ..., (q_{|\mathcal{D}^p|}^p, e_{|\mathcal{D}^p|}^p)\}$ to train a MWP model that can make prediction on a test dataset $\mathcal{D}^q$.

2.1 Split Data with DBCA

Distribution-based compositional assessment (DBCA) [15] is proposed to create compositional splits for KBQA task. Each data is represented using a graph consisting of atoms (nodes) and compounds of atoms. Then based on DBCA, a greedy algorithm is applied to assign each data to training and test datasets. The detailed description of DBCA for split could be found in Appendix B.

When it comes to MWPs, each expression of MWPs is represented using an expression tree [41], which can be treated as a specific graph. The leaf nodes are quantities and they are connected via mathematical operators. However, the lack of semantics of quantities leads to inapplicability of the above divergence measurement. As shown in Figure 2, (a) and (b) describe the different scenarios but their graphs are exactly the same. (a) and (c) describe the similair scenarios but with different quantities, which leads to different graphs. Therefore, we enrich the representation of quantity in the expression tree by its contexts. Here, we represent quantities via cluster indexes derived from k-means clustering algorithm over contextual TF-IDF vectors of quantities. In Figure 2, after representing the trees featured with context information, (a) and (c) will have the same graphs. Next, we follow the

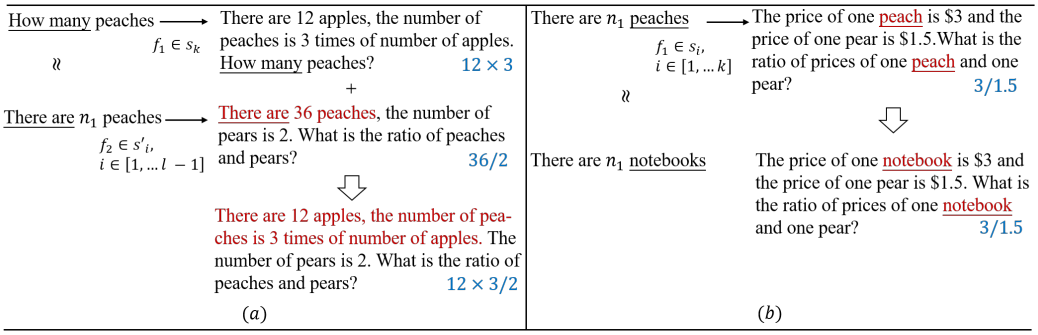


Figure 3: Two examples of data augmentation protocol on MWPs via operation 1 and operation 2 in sub-figure (a) and (b), respectively. Fragments are labeled with underline and substituent parts in the synthetic MWPs are in red color. Expressions are annotated with blue color.

traditional DBCA and the greedy algorithm to generate realistic compositional splits. More detailed description could be found in Appendix B.

2.2 Synthesize Data to Isolate Effect

To understand how MWP solvers respond to different compositions precisely, we also synthetically construct data via rules. In Appendix C, we provide detailed description of this data construction.

3 Data Augmentation for MWP Solvers

Previous studies [13, 1] have conducted discussions on data augmentation, which has proven to be effective in improving the robustness of models. In this manuscript, we propose an iterative data augmentation method that aims to generate more compositional variation without involving external data and human annotation. Inspired by GECA [1], we discover the alternative text fragments and substituent the text fragments in existing MWPs to synthesize new MWPs. We further improve the quality of the generated data by measuring the confidence of new MWPs via a well-trained MWP solver. The entire procedure is displayed in Algorithm 1. As we can see, this algorithm is designed with an iterative framework, where a *Data Augmentation Protocol* takes charge of generating new MWPs and a *Data Ranker* plays the role of judging the quality of the generated MWPs. After T rounds of iteration, the augmented data will fill the compositional gap which is leveraged to train a robust MWP model.

Algorithm 1 Data Augmentation Algorithm

Input: initial training data set $\mathcal{D}^p$, max iteration T
for *i in T* **do**
  $\mathcal{D}^* = \emptyset$
  for *q in* $\mathcal{D}^p$ **do**
    $\mathcal{D}^p_q$ = **DataAugmentationProtocol**(q)
    $\mathcal{D}^* = \mathcal{D}^* \cup \mathcal{D}^p_q$
  end
  $\mathcal{D}^*$ = **DataRanker**($\mathcal{D}^*$)
  $\mathcal{D}^p = \mathcal{D}^p \cup \mathcal{D}^*$
end
Return: augmented training data set $\mathcal{D}^p$

Figure 3 show two example of our data augmentation protocol. We refer to more example analysis in Appendix D. We now formally describe the operations in the data augmentation protocol. Recall each MWP consists of multiple sentences $q = \{s_1, ..., s_k\}$. A fragment is a set of non-overlapping spans of s. An environment is s with a text fragment removed. We denote fragment and environment as f and s/f, respectively. If there are environments $x = s_1/f_1$ and $y = s_2/f_2$ with $x = y$, (f_1, f_2) is a pair of alternative fragments: 1) **Operation 1**. Given $q = \{s_1, ..., s_k\}$ and $q' = \{s'_1, ..., s'_l\}$ with e and e', respectively. If $f_1 \in s_k$ and $f_2 \in s'_i$, where $i \in [1, ..., l-1]$, we could form a new MWP $q' = \{s'_1, ..., s'_{i-1}, s_1, ..., s_{k-1}, s'_{i+1}, ..., s'_l\}$. The expression of this MWP is e' modified by replacing the quantity in s'_i with e; 2) **Operation 2**. Given $q = \{s_1, ..., s_k\}$ with e, if $f_1 \in s_i$ where $i \in [1, ..., k-1]$, we could form a MWP $q' = \{s'_1, .., s'_i, ..., s'_k\}$ where s'_i is obtained via replacing all the f_1 in s_i with f_2. The expression of this MWP is still e. For each $q \in \mathcal{D}^p$, we generate a set of new MWPs, via either operation 1 or operation 2. These newly generated MWPs will form a set $\mathcal{D}^*$.

Methods	Math23K I.I.D.	Math23K ComDiv	Math23K Rel. Gap	MAWPS I.I.D.	MAWPS ComDiv	MAWPS Rel. Gap	SD Valid	SD Test	SD Rel. Gap
LSTM	67.4	47.4	0.42	74.5	36.5	1.04	93.7	7.8	11.0
Transformer	58.0	40.3	0.44	72.6	40.1	0.81	91.0	6.6	12.8
BERTGen	68.1	43.0	0.58	70.7	43.0	0.64	96.9	9.6	9.09
GPT-2	70.4	49.7	0.42	62.2	31.9	0.95	95.1	8.3	10.5
MathEN	66.5	46.9	0.42	76.4	40.9	0.87	94.6	7.5	11.7
GTS	71.5	51.2	0.39	76.9	48.4	0.59	**96.9**	7.9	11.3
MathEN+DA	67.3(+0.8)	47.5(+0.6)	0.42	77.6 (+1.2)	42.4(+1.5)	0.83	95.5 (+0.9)	17.4(+9.9)	4.5
GTS+DA	**72.1**(+0.6)	**52.0**(+0.8)	**0.38**	**78.1**(+1.2)	**49.8**(+1.4)	**0.57**	96.4(-0.5)	**23.5** (+15.6)	**3.1**

Table 1: Answer accuracy of four groups of methods on different data sets. We show the accuracy of test data for different splits of Math23K and MAWPS, the accuracy of validation and test data for SD data set. To show the performance gap between normal and compositional splits, we compare the results of Math23K and MAWPS with I.I.D. and ComDiv splits, results of SD with validation and test sets, respectively. **Rel. Gap** denotes $Acc_{I.I.D.}/Acc_{ComDiv} - 1$. The numbers in red font denote improvement brought by data augmentation.

.

4 Experiments

We create compositional challenges deriving from two datasets: Math23K dataset from Wang [38] and MAWPS dataset from Kedziorski [17]. As a result, we obtain the following three compositional data sets: **Math23K-ComDiv**, **MAWPS-ComDiv**, and a synthetic dataset **SD**. The statistics of all datasets and the detailed data processing could be found in Appendix E.1

Comparable Methods. The methods tested on compositional challenges of MWPs are mainly categorized into four families: (1) General encoder-decoder models, i.e., **LSTM** [35] and **Transformer** [10]. (2) Strong MWP solving methods, i.e., **MathEN** [36] and **GTS** [39]. (3) Pretrained models i.e., **BERTGen** [4] and **GPT-2** [29], which have been suggested to improve general compositional generalization [27]. (4) MathEN and GTS collaborating with our proposed data augmentation method (refer to as **MathEN+DA** and **GTS+DA**), which aims to improve compositional generalization.

4.1 Results

Table 1 displays the results of all tested methods. Based on the table, we have the following observations: (1) For SD data set, we observe a huge performance gap between validation set and test set. A similar performance gap appears between I.I.D. split and ComDiv split on both Math23K and MAWPS. This indicates that all families of methods are vulnerable to compositional challenges MWP solving. Among these three datasets with compositional challenges, SD dataset causes the biggest loss. This could be explained by the nature of big compound divergence of SD data. (2) Comparing the first three sections, in terms of the most robust method to compositional challenges, there is no absolute agreement on all the data sets. Nevertheless, GTS shows relatively good compositional generation as it has the lowest Rel.Gap on Math23K and MAWPS. This may because the tree-structured neural module which outputs an expression tree is more powerful when generalizing to unseen compositions. (3) Our data augmentation could provide positive effect to MathEN and GTS methods. This indicates that data augmentation is effective in improving the compositional generalization capability of general MWP methods. Meanwhile, data augmentation shows the significant contribution on SD dataset with 9.9 and 15.6 percentage points improvement to MathEN and GTS, respectively. This implies it indeed plays a key role in bridging the gap of compositions. It's worth noting that data augmentation provides the largest performance gain 15.6% to GTS model on SD test set. It may because that data augmentation includes more lexical and structural variations for training and tree-structured neural network has a better inductive bias to capture them, which is also verified in prior work [25]. We also provide further analysis, including accuracy with increasing compound divergence, effect of different composition Types, ablation Study of data augmentation, effect of iteration time, and case study of augmented data, which could be found in Appendix F.

5 Conclusion

In this manuscript, we investigated compositional generalization in MWP solving task. We created three compositional challenges on MWPs and evaluated a set of methods. We observed most of the methods suffered from a big loss of performance. We further proposed an iterative data augmentation method and it has proven to be effective in improving the compositional generalization of general methods. While this problem is still open to be explored, our study provides some insights for proposing new solutions in the future research.

References

[1] Jacob Andreas. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, 2020.

[2] Dzmitry Bahdanau, Harm de Vries, Timothy J. O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019.

[3] Rahma Chaabouni, Eugene Kharitonov, Diane Bouchacourt, Emmanuel Dupoux, and Marco Baroni. Compositionality and generalization in emergent languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4427–4442. Association for Computational Linguistics, 2020.

[4] Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, 2020.

[5] Ting-Rui Chiang and Yun-Nung Chen. Semantically-aligned equation generation for solving and reasoning math word problems. 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)*, pages 2656–2668, 2019.

[6] J. Chung, P. Kannappan, C. T. Ng, and P. Sahoo. Measures of distance between probability distributions. *Journal of Mathematical Analysis and Applications*, 138:280–292, 1989.

[7] Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, 2018.

[8] Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating nlp models via contrast sets. *arXiv preprint*, 2020.

[9] Yu Gu, Sue Kase, Michelle T. Vanni, Brian M. Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of The Web Conference 2021*, 2021.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. Smart: A situation model for algebra story problems via attributed grammar. In *arXiv preprint arXiv:2012.14011*, 2020.

[12] Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814, 2017.

[13] Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, 2016.

[14] Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020.

[15] Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. Measuring compositional generalization: A comprehensive method on realistic data. In *8th International Conference on Learning Representations*, 2020.

[16] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, 2020.

[17] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, A. Sabharwal, Oren Etzioni, and S. D. Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

[18] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 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, 2016.

[19] Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 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, November 2021. Association for Computational Linguistics.

[20] Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 271–281, 2014.

[21] Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *35th International Conference on Machine Learning*, pages 4487–4499, 2018.

[22] Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 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, 2019.

[23] Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. Reverse operation based data augmentation for solving math word problems, 2021.

[24] Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 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*, pages 2370–2379, 2019.

[25] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, 2019.

[26] Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, 2016.

[27] Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. Improving compositional generalization in semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2482–2495, 2020.

[28] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 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, 2021.

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[30] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, 2015.

[31] Subhro Roy and Dan Roth. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, page 3082–3088, 2017.

[32] Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems*, pages 19861–19872, 2020.

[33] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? *arXiv preprint arXiv:2010.12725*, 2020.

[34] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, 2015.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

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

[37] Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. pages 5545–5552, 2018.

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

[39] Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5299–5305, 2019.

[40] Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020.

[41] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 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, 2020.

[42] Yanyan Zou and Wei Lu. Quantity tagger: A latent-variable sequence labeling approach to solving addition-subtraction word problems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5246–5251, 2019.

A Related Work

A.1 Math Word Problem Solving

Early work on math word problem solving task can be categorized into statistical machine learning based methods [20, 26, 31, 42] and semantic parsing based methods [34, 30, 12]. The hand-crafted features are collected to represent MWPs and pre-defined templates are leveraged to generate answers. However, they cannot be applied to large-scale datasets. Recently, neural networks have been widely applied to solve MWPs. Seq2seq models were first leveraged to directly transform MWP text sequence to expression sequence [38], which are still the mainstream MWP solvers until now. Li [22] borrowed the idea from Transformer and proposed multi-head attention to model different types of features of MWPs. Many methods managed to encode richer information, they proposed different ways to pre-process MWPs, such as number mapping [38] and graph construction [41]. To capture the structural information of expressions, more advanced work [36, 5, 24, 39] proposed to decode an expression with implicit or explicit tree structure. Besides, there are some other studies focusing on knowledge distillation [40], weak supervision [11] and data augmentation [23] for MWP solving.

A.2 Compositional Generationation

Compositional generalization gains much attention from researchers recently. People have shown that neural network-based methods always encounter tremendous loss when facing the compositional challenges [8, 14, 27]. Different datasets have therefore been introduced to support the compositional generalization research on Language-driven Navigation [21, 32], Question Answering [2, 15, 9], Emergent Languages [3] and text2SQL [7]. However, there is a lack of study of compositional generalization on MWP solving task. Hong [11] briefly discussed the OOD challenge by splitting the MWP data based on the length of MWPs. Kumar [19] discussed the robustness of MWP solvers under the adversarial attack of question reordering and sentence paragraphing. Patel [28] included carefully chosen text variations to MWPs for more robust evaluation of methods. The focus of them is more about length divergence and text perturbation rather than compositional generalization.

B Split Data with DBCA

As introduced in the Sec. 1, existing work [15, 33] has proposed their method based on distribution-based compositional assessment (DBCA) to create compositional splits for KBQA task, which could be summarized as follows:

- Each data is represented using a graph, where the nodes are considered as *Atoms* and the rule applications on the graph are treated as *Compounds*.
- A distribution-based compositional assessment [15] is utilized to measure the divergence: $DBCA(\mathcal{D}^p, \mathcal{D}^q) = 1 - \mathcal{C}_\alpha(\mathcal{PQ})$, where $\mathcal{C}_\alpha(\mathcal{PQ}) = \sum_i p_i^\alpha q_i^{1-\alpha} \in [0, 1]$ is the Chernoff coefficient [6], $\mathcal{P}$ and $\mathcal{Q}$ are distributions deriving from training and test sets, respectively. In more detail, they obtain atom and compound distributions based on their frequencies, and then compute *Atom Divergence* $DBCA^a$ and *Compound Divergence* $DBCA^c$, respectively. As we can see, the atom divergence measures the constituent difference and compound divergence measures the compositional difference of a dataset.
- Starting from a data pool $\mathcal{D}$, they apply a greedy algorithm to assign each data to form $\mathcal{D}^p$ and $\mathcal{D}^q$, the objective of which is to maximize $DBCA^c$ with the control of an upper bound of $DBCA^a$.

The detailed algorithm could be found in original work [33].

When it comes to MWPs, each expression of MWPs could be represented using an expression tree [41], which can be treated as a specific graph. The leaf nodes are quantities and they are connected via mathematical operators. However, the lack of semantics of the quantities leads to inapplicability of the above divergence measurement. As shown in Figure 2, (a) and (b) describe the different scenarios but their graphs are exactly the same. (a) and (c) describe the similair scenarios but with different quantities, which leads to different graphs. Therefore, we enrich the representation of quantity in the expression tree by its contexts. Specifically, we extract tokens within a window slide centered on a target quantity in a MWP to represent the quantity. Then, we represent the quantities via

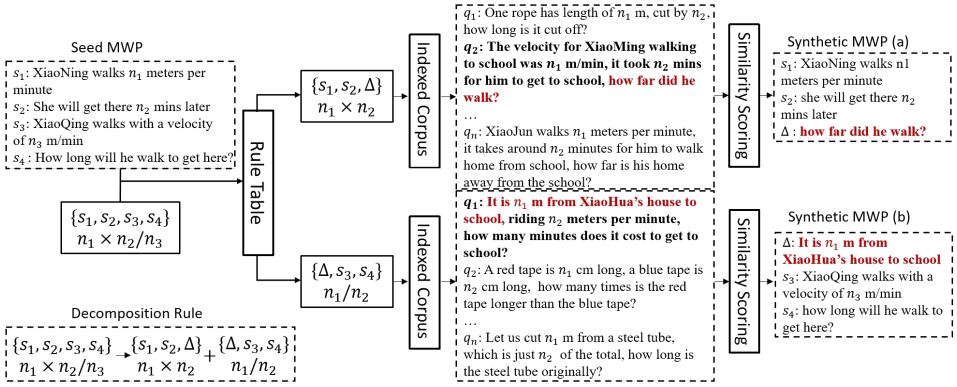


Figure 4: One example of generating synthetic MWPs with a seed MWP via a decomposition rule. The selected MWP in the corpus is shown in bold line and the instantiate sentence of Δ is annotated with red color.

TF-IDF vectors and perform k-means clustering algorithm based on the vector-based representations, which assigns a cluster index to each quantity[2]. As a result, the quantities within similar contexts are labeled with the same cluster indexes and we treat such quantities annotated with cluster indexes as atoms. In Figure 2, after representing the trees featured with context information, (a) and (c) will have the same graphs. Next, $DBCA^a$ is derived from the frequency of cluster indexes after traversing the expression tree. To obtain $DBCA^c$, we exhaustively search sub-expression trees in a complete expression tree. Each sub-expression tree consists of a left component, an operator node and a right component, where a component is either a sub-expression tree or an atom in the expression tree. Afterwards, we follow the traditional distribution-based assessment and the greedy algorithm to generate realistic compositional splits.

C Synthesize Data to Isolate Effect

Data splitting method results in datasets differing in compositions with a fuzzy boundary. Since our goal is to understand how MWP solvers respond to different compositions precisely, to disentangle compositional gaps arising from new expression templates or new combinations of templates and constituents, we synthetically construct data via rules.

As the preliminary step, we pre-define a *Rule Table* [3], which contains the rules that we follow to synthesize data. Each rule states that given a MWP with an expression template, it could be transformed into new MWP(s) with certain expression template(s). For example, the MWP (d) in Figure 1 could be represented by $\{s_1, s_2, s_3, s_4\}$ with expression template $n_1 \times n_2/n_3$, where the quantity in the expression is replaced by the position index [37]. Following one rule, we could decompose it into $\{s_1, s_2, \Delta, s_4\}$ with template $n_1 \times n_2$ and $\{\Delta, s_3, s_4\}$ with template n_1/n_2, where Δ indicates a placeholder to be instantiated. We further generate an *Indexed Corpus* by indexing a set of initial MWPs with their templates as keys and MWPs as values, which could be utilized to instantiate the placeholders. Moreover, we define a *Similarity Scoring* function to measure the semantic similarity between MWPs. Specifically, we extract a collection of nouns, verbs and measure units from the MWPs using existing toolkits. When we measure the similarity of two MWPs, we compare the token-level overlap. A large similarity value indicates the MWPs share similar constituents. Next, we introduce how we synthesize data with the two types of compositions:

- **Decomposition**. We treat each decomposable $q \in \mathcal{D}$ as the seed MWP, where $\mathcal{D}$ is a data pool. We employ applicable decomposition rules from the rule table to generate two uninstantiated MPWs. We retrieve all MWPs from the indexed corpus with their

[2]We have also tried other methods such as Word2vec, BERT to represent the context. They cannot outperform TF-IDF vectors even though they have more expressive architectures. It could be explained that TF-IDF vectors are derived from the MWPs domain so it contains more domain knowledge while other models involve more general knowledge.

[3]The complete table and more implementation details are displayed in Appendix

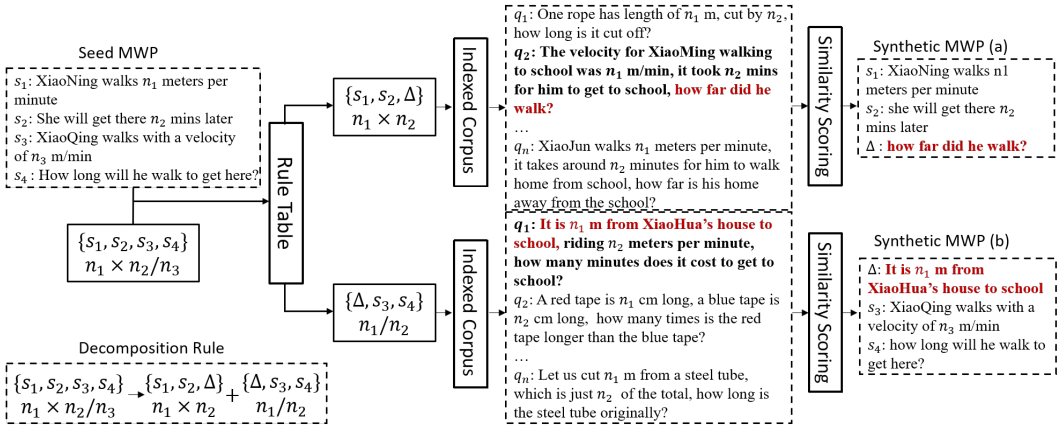


Figure 5: One example of generating synthetic MWPs with a seed MWP via a decomposition rule. The selected MWP in the corpus is shown in bold line and the instantiate sentence of Δ is annotated with red color.

corresponding templates, select the MWP which has the highest similarity score to the seed MWP and instantiate Δ with corresponding sentence of the selected MWP. This step will generate two synthetic MWPs $\{q^{'}, q^{''}\}$ specific to q. To illustrate the above procedure better, we display an example in Figure 5. Eventually, we allocate (q, e) into test set and $\{(q^{'}, e^{'}), (q^{''}, e^{''})\}$ into training set to form $\mathcal{D}^q$ and $\mathcal{D}^p$, respectively. It is worth noting that sometimes the selected MWP $q^{'}$ or $q^{''}$ has similarity score value to q as 0, in this case, we abandon it. This is to ensure the similar constituent distribution between $\mathcal{D}^p$ and $\mathcal{D}^q$.

- **Reformulation**. For each applicable $q \in \mathcal{D}$, we employ reformulation rules to generate one uninstantiated MWP and instantiate placeholder Δ as illustrated as above to obtain a synthetic MWP q'. If for any existing $q'' \in \mathcal{D}^p$ which has $e'' = e'$ and the similarity score to q' is 0, we include (q, e) into $\mathcal{D}^p$ and (q', e') into $\mathcal{D}^q$. Otherwise, we abandon it. This is because we would like to only include the MWP of which the combination of templates and constituents have not been seen during training into the test set.

After above procedure, we obtain the synthetic data sets $\mathcal{D}^p$ for training and $\mathcal{D}^q$ for testing. $\mathcal{D}^p$ includes all the MWPs having simple templates as their expressions, $\mathcal{D}^q$ contains all the MWPs which are either new compounds of the simple expression templates or new combinations of existing constituents and simple expression templates. As synthesized MWPs always share constituents with seed MWP, the constituent difference between training and test sets is well-controlled and the only variation is the compositionality.

C.1 Detailed Implementation for Synthesizing Data

To synthesize data, we manually defined a *Rule Table* with 13 rules in total to generate synthetic data which are displayed in Table 2. For example, the MWP (d) in Figure 5 could be represented by $\{s_1, s_2, s_3, s_4\}$ with expression template $n_1 \times n_2/n_3$, where the quantity in the expression is replaced by the position index [37]. Following one rule, we could decompose it into $\{s_1, s_2, \Delta, s_4\}$ with template $n_1 \times n_2$ and $\{\Delta, s_3, s_4\}$ with template n_1/n_2, where Δ indicates a placeholder to be instantiated. We further generate an *Indexed Corpus* by indexing a set of initial MWPs with their templates as keys and MWPs as values, which could be utilized to instantiate the placeholders. Moreover, we define a *Similarity Scoring* function to measure the semantic similarity between MWPs. Specifically, we extract a collection of nouns, verbs and measure units from the MWPs using existing toolkits. When we measure the similarity of two MWPs, we compare the token-level overlap. A large similarity value indicates the MWPs share similar constituents.

Then we synthesize data as describedC. For each applicable MWP, we first decompose or reformulate it with rule table, then we find all MWPs with corresponding templates via indexed corpus. We further rank these MWPs with similarity scoring and instantiate Δ with the sentence in the best matched

	ID	Seed MWP	Synthetic MWP(s)	
Decomposition	1	$\{s_1, s_2, s_3\}$ $n_1 \times (1 - n_2)$	$\{s1, \Delta, s_3\}$ $n_1 \times n_2$	$\{s_2, \Delta\}$ $1 - n_1$
	2	$\{s_1, s_2, s_3, s_4\}$ $n_1 \times n_2 / n_3$	$\{s_1, s_2, \Delta\}$ $n_1 \times n_2$	$\{\Delta, s_3, s_4\}$ n_1 / n_2
	3	$\{s_1, s_2, s_3, s_4\}$ $n_1 \times n_2 + n_3$	$\{s_1, s_2, \Delta\}$ $n_1 \times n_2$	$\{\Delta, s_3, s_4\}$ $n_1 + n_2$
	4	$\{s_1, s_2, s_3, s_4\}$ $n_1 \times n_2 \times n_3$	$\{s_1, s_2, \Delta\}$ $n_1 \times n_2$	$\{\Delta, s_3, s_4\}$ $n_1 \times n_2$
	5	$\{s_1, s_2, s_3\}$ $n_1 / (1 - n_2)$	$\{s_1, \Delta, s_3\}$ n_1 / n_2	$\{s_2, \Delta\}$ $1 - n_1$
	6	$\{s_1, s_2, s_3, s_4\}$ $(n_1 - n_2) / n_3$	$\{s_1, s_2, \Delta\}$ $n_1 - n_2$	$\{\Delta, s_3, s_4\}$ n_1 / n_2
	7	$\{s_1, s_2, s_3, s_4\}$ $(n_1 + n_2) \times n_3$	$\{s_1, s_2, \Delta\}$ $n_1 + n_2$	$\{\Delta, s_3, s_4\}$ $n_1 \times n_2$
	8	$\{s_1, s_2, s_3, s_4\}$ $n_1 / n_2 \times n_3$	$\{s_1, s_2, \Delta\}$ n_1 / n_2	$\{\Delta, s_3, s_4\}$ $n_1 \times n_2$
	9	$\{s_1, s_2, s_3, s_4\}$ $n_1 \times n_2 - n_3$	$\{s_1, s_2, \Delta\}$ $n_1 \times n_2$	$\{\Delta, s_3, s_4\}$ $n_1 - n_2$
Reformulation	10	$\{s_1, s_2, s_3\}$ $n_1 + n_2$	$\{s_1, s_2, \Delta\}$ $n_1 - n_2$	
	11	$\{s_1, s_2, s_3\}$ $n_1 - n_2$	$\{s_1, s_2, \Delta\}$ $n_1 + n_2$	
	12	$\{s_1, s_2, s_3\}$ $n_1 \times n_2$	$\{s_1, s_2, \Delta\}$ n_1 / n_2	
	13	$\{s_1, s_2, s_3\}$ n_1 / n_2	$\{s_1, s_2, \Delta\}$ $n_1 \times n_2$	

Table 2: The rule table used in our work. The middle section contains all the rules for decomposition and the third section contains all the rules for reformulation.

MWP. We display an example of generating synthetic MWPs with a seed MWP via a decomposable rule in Figure 5.

C.2 Rule Table for Synthesizing Data

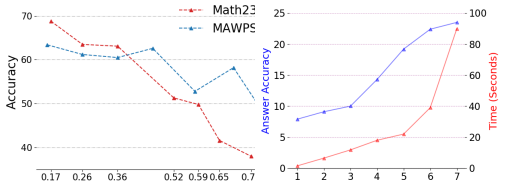
We manually defined 13 rules in total to generate synthetic data (Sec. C) which are displayed in Table 2.

D Example Analysis of Our Data Augmentation

We first describe our data augmentation protocol which is an improved version of GECA. Two text fragment is defined as *Alternative* if they occur in some common environments. With the alternative fragments, we define the following operations to involve new compositions in a MWP: (1) If the first fragment is from the last sentence of a MWP, which is the question sentence, and the second fragment is not from the last sentence of another MWP, we use the sentences excluding the last sentence of the first MWP to substituent the corresponding sentence of the second MWP. In the example of Figure 3 (a), "how many" and "there are" are alternative fragments as they appear in the same environment "peaches". Given a MWP querying "how many", we could leverage its narrative description to substituent the sentence "there are 36 peaches" in another MWP to form a new MWP. (2) If both the fragments are not from the last sentences of MWPs, we replace all the first fragment in a MWP with the second fragment. In the example of Figure 3 (b), "peach" and "notebook" is a pair of alternative fragments. When there is a MWP with fragment "peach", we can generate a new MWP by replacing all the "peach" with "notebook". As we can see, the first operation involves more diversity to the expression structure and generates MWPs with novel expression templates. The second operation helps to generate MWPs with novel combinations of expression templates and constituents.

	Split	NR	LR	$DBCA^a$	$DBCA^c$
Math23K	I.I.D.	18543/4619	0.99	0.00	0.20
	ComDiv	18543/4619	1.01	0.00	0.69
MAWPS	I.I.D.	1899/474	1.01	0.00	0.09
	ComDiv	1899/474	0.91	0.01	0.77
SD	ComDiv	1078/745	0.76	0.01	0.82

Table 3: Statistics of datasets. **NR**, and **LR** denote Number Ratio and text Length Ratio between training and test sets, respectively. $DBCA^a$ and $DBCA^c$ are atom and compound divergences, respectively.



(a) (b)

Figure 6: (a) Performance changes of MathEN with the increase of $DBCA^c$ on Math23K and MAWPS. (b) Evolution of answer accuracy (blue) and average training time per epoch (red) of GTS on SD dataset with the increase of iteration time.

E Experimental Setting

E.1 Compositional Data Sets

These two datasets are both large-scale MWP datasets that are commonly used in existing work. We utilize NLTK[4] as the toolkit to extract contextual features through tokenization, POS tagging and dependency parsing. For data splitting, scikit-learn tool[5] is employed to implement TF-IDF. We choose k value from range $\{5, 10, 20\}$ based on the highest $DBCA^c$ it could achieve after splitting. Following previous implementation[6], α in DBCA is set as 0.1 and 0.5 for $DBCA^a$ and $DBCA^c$, respectively.

E.2 Implementation Details.

All the experiments are implemented by PyTorch on Nvidia V440.64.00-32GB GPU cards. We implement the existing work by leveraging their official code or following their methodology description. Regarding general encoder-decoder models, we set the same embedding size and hidden size as MathEN. Regarding pretrained baselines, we adopt base BERT and base GPT-2. Regarding MWP baselines and MWP baselines with data augmentation, we keep hyperparameters the same as original manuscripts[7]. For each type of data splits, we randomly allocate 20% data from training to do validation. In terms of the data augmentation method, the MWP solvers well trained with the augmented data in the last round of iteration are used to rank the data. We set the maximum iteration time T as 7 and keep the top 20% augmented MWPs for each iteration.

	SD	
	Structural	**Lexical**
MathEN	3.0	46.3
GTS	2.9	51.3
MathEN+DA	14.2	47.5
GTS+DA	16.8	63.8

Table 4: Results on partition of SD dataset with Structural and Lexical generalization separately.

	Math23K-ComDiv	**MAWPS-ComDiv**
MathEn+DA	47.5	42.4
- DataRanker	46.6	41.4
- Operation 1	47.3	41.8
- Operation 2	47.2	41.6

Table 5: Ablation study of data augmentation with MathEn on Math23K-ComDiv and MAWPS-ComDiv datasets.

F Further Analysis

F.1 Accuracy with Increasing Compound Divergence

To show the influence of compound divergence described in Sec. 2.1, we display Figure 6 (a). As we can see, the performance of MathEN on both Math23K and MAWPS datasets decreases dramatically with the increase of compound divergence $DBCA^c$. This indicates the performance is sensitive to the change of compound divergence, which verifies that the compound divergence is indeed an effective indicator for measuring the compositional challenge of MWPs. Therefore, it is reasonable for us to split MWPs data and obtain the compositional data sets with $DBCA^c$ performing as the measurement criteria.

F.2 Effect of Different Composition Types

SD dataset that we introduced in Sec. C is created for isolating the effect of different types of compositions. Decomposition and reformulation rules create data requiring structural and lexical generalization, respectively. We display Table 4 to discuss the effect of different rules. From the table, we observe that new compound expression templates bring much more difficulties to models than new combinations of expression templates and constituents. It is easy to understand as template inferring requires a deep understanding towards the expression. We also observe that data augmentation benefits more to MWPs with new compound expression templates.

F.3 Ablation Study of Data Augmentation

To further investigate the mechanism of the proposed data augmentation method, we remove the data ranker, operation 1 and operation 2 in turn and test MathEn+DA on the Math23K-ComDiv and MAWPS-ComDiv datasets. We show the results in Table 5. As we can see, removing any of them causes the decrease of results. This indicates that data ranker, operation 1 and operation 2 all contribute to the good effect of data augmentation method. Operation 1 and operation 2 generate different compositions to fill the gap. Data ranker prevents the model from the hurt of low-qualified augmented data.

[4] `https://www.nltk.org/`

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[6] `https://github.com/google-research/language/tree/master/language/nqg`

[7] We show detailed hyperparameters in Appendix.

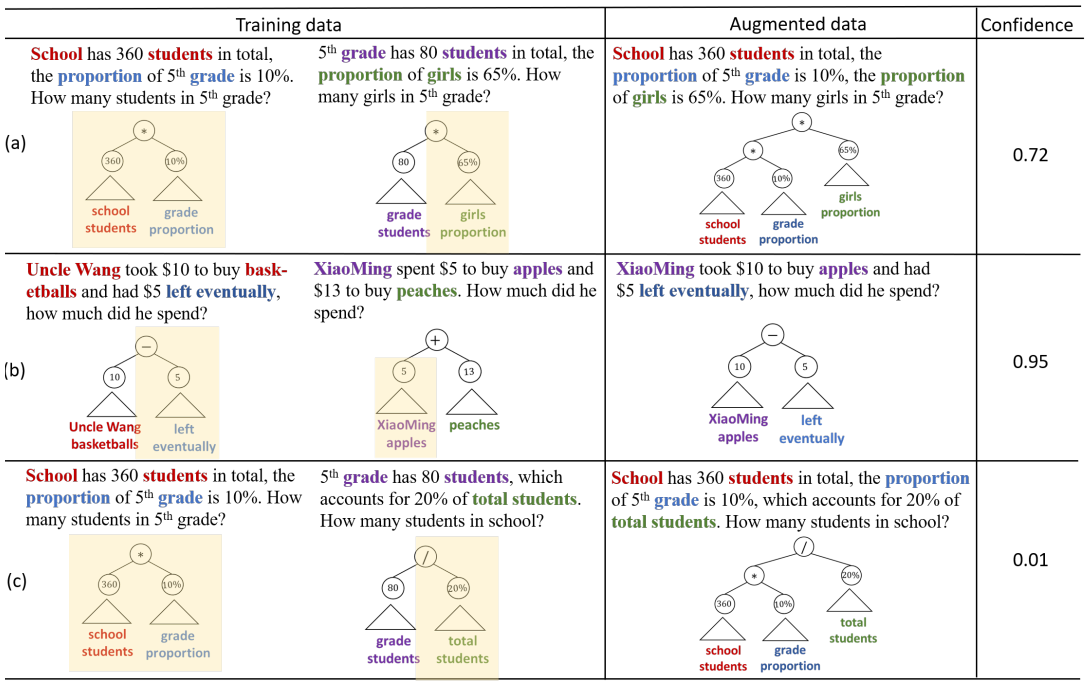


Figure 7: Three augmented examples produced via our method. The original training examples of Math23K is shown at the left column, corresponding augmented data is shown at the middle column and the confidence score for the augmented data is shown at the right column. Each MWP is associated with its expression tree. The context of a quantity in MWPs is highlighted with colors and the partial expression trees forming the augmented data is highlighted with yellow boxes.

F.4 Effect of Iteration Time.

We depict the curve of data augmentation effect with the influence of iteration time in Figure 6 (b). As we can see, with the increase of iteration time, the answer accuracy of GTS on SD dataset increases gradually. It is because more iteration indicates more augmented data is generated to fill the compositional gap. But when iteration comes to 6, the increase becomes less significant. Meanwhile, we notice that with the increase of iteration, the average time for each training epoch becomes larger. It is because that the number of augmented data increases exponentially, which leads to a large time cost for training the model. Therefore, it is a trade-off to select a suitable iteration time with the consideration of both performance gain and time cost.

F.5 Case Study of Augmented Data

We display some augmented data in Figure 7. As we can see, the expression tree of augmented data is a novel combination of the expression trees of training data. The math word problems are the corresponding narratives of the expression trees. Example (a) shows that data augmentation method generates a novel combination of two familiar structures. The training data contains example of the structure $n_1 \times n_2$. But the combination $n_1 \times n_2 \times n_3$ has never been observed in training. The augmented data fills this gap. After adding the augmented data into the training set, such combination rules will be learned by models. Example (b) shows that data augmentation generates a novel combination of a familiar primitive and a familiar structure. The training data contains example of the structure $n_1 - n_2$ but the generalization concerns the constituents "XiaoMing", "apples" has never been observed in the first position of $n_1 - n_2$. The augmented data fills this gap by including more lexical variations into the training data and thus improve the lexical generalization of models. Example (c) is another augmented example of structural generalization. Even though it is semantically correct, it is logically wrong. When a data ranker generates its math expression, and the confidence score is very low. Therefore, this is not a good augmented example and our method excludes it from the training set.

F.6 Human Evaluation of Augmented Data

In order to examine the quality of augmented data and understand how the augmented data improve general MWP methods, we invite two people to evaluate the quality of augmented MWPs. An augmented MWP will be labeled as qualified by human evaluators if it satisfies grammatical correctness, fluency, and logical consistency. Otherwise, it will be labeled as unqualified. For simplicity, we randomly sample 100 augmented MWPs and ask human evaluators to judge the quality of each MWP. We then calculate the agreement between evaluators using Cohen's k coefficient. Finally, the percentage of qualified augmented examples from two evaluators are 83% and 84% respectively, and their Cohen's k coefficient is 0.71, which indicates the data augmentation indeed generates a decent number of high-qualified data examples to strength the training procedure.