Improving Equation Set Problems with Label Augmentation

Anonymous ACL submission

Abstract

Math word problems solving has received considerable attention from many NLP researchers. Inspired by the encoder-decoder structure, they created a series of neural network models to solve arithmetic word problems and equation set problems. However, these encoder-decoder models used the ground truth as the only generation target, resulting in shallow heuristics to generate expressions. In this paper, we propose a simple and effective label augmentation method for equation set problems. Specifically, we transform the ground truth into several equivalent labels by normalization rules, and these new labels will be used as additional generation targets for model training. Experimental results on the English dataset DRAW1K and Chinese dataset HMWP show that the label augmentation method has at most 4.5% improvement over the state-of-the-art (SoTA) models.

Problem:	A number added to **6** is equal to **30** less than **four** times the number. What is the number?
Equation:	$x + 6 = 4x - 30$
New Label:	$x = (6 + 30) \div (4 - 1)$
Answer:	12
Problem:	What is the number of chickens and rabbits in a cage, with **20** heads on the top, and **50** legs on the bottom?
Equation:	$\begin{cases} x + y = 20 \\ 2x + 4y = 50 \end{cases}$
New Label:	$\begin{cases} x = (4 * 20 - 50) \div (4 - 2) \\ y = (50 - 2 * 20) \div (4 - 2) \end{cases}$
Answer:	$(15, 5)$

Table 1: The examples of equation set problems.

1 Introduction

Math Word Problems (MWP) solving is a popular NLP task, which requires a solver to provide a solvable expression towards a mathematical question by understanding the semantics and logic of its narrative description (Zhang et al., 2020a). According to the form of expressions, MWP can be divided into arithmetic word problems with one-unknown variable and equation set problems where one/multiple-unknown variables are included in the expression (Qin et al., 2020). Table 1 shows two examples of equation set problems.

To solve MWP with deep learning, many researchers (Wang et al., 2017; Xie and Sun, 2019; Zhang et al., 2020b) applied the encoder-decoder structure. They used recurrent neural networks or pre-trained language models to encode the problem texts and generated expressions by sequence-based decoders or tree-based decoders. Although these neural network models achieved promising results for arithmetic word problems, this training paradigm that directly takes ground truth as the only generation target has significant pitfalls. Some work (Patel et al., 2021; Kumar et al., 2021) provided the evidence that these methods rely on shallow heuristics to generate expressions, which leads to the failure of generalization.

To address the above issue, recent studies (Liang and Zhang, 2021; Shen et al., 2021; Huang et al., 2021; Li et al., 2021; Qin et al., 2021) adopted a multi-task framework to increase the difficulty of model training. Unlike they focused on designing auxiliary tasks on arithmetic word problems, we propose a label augmentation method for equation set problems in this paper. Specifically, we transform the ground truth into several equivalent labels by some normalization rules, and these augmented labels will be used as additional generation targets for model training. The motivation of our approach comes from two aspects: (1) Mathematical expressions have various forms naturally since the existence of commutative laws; (2) For equation set problems, there are different solutions from

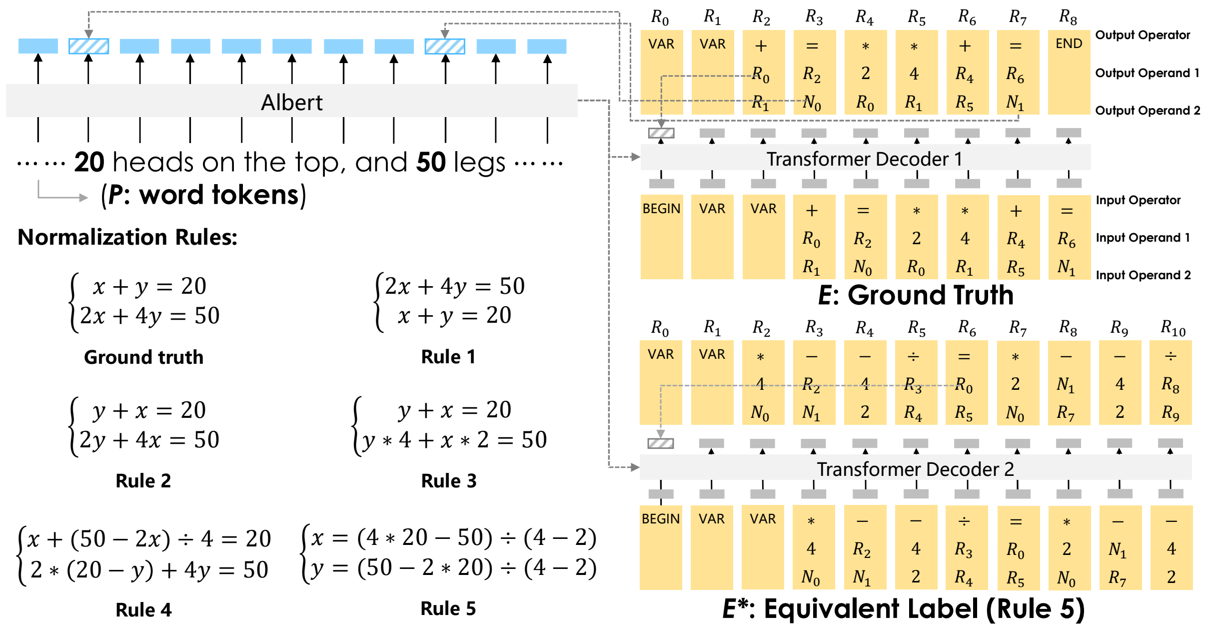

Figure 1: The framework of our model, which follows the encoder-2decoders architecture. One decoder aims to generate original labels and another one aims to fit the nature of augmented labels.

the ground truth. For example, we can obtain new solutions for the two problems in Table 1.

Hence, we summarize five normalization rules to produce these equivalent labels, which are: (1) equation swapping; (2) variable substitution; (3) commutative law; (4) new solution 1; (5) new solution 2. These equivalent labels produce varying degrees of difference from ground truth (difference: Rule 1<2<3<4<5). To facilitate the encoder-decoder structure adapt to the augmented labels, we replace the traditional single decoder with two decoders in parallel, where one aims to generate original labels and another one aims to fit the nature of augmented labels.

The main contributions of this paper are:

- We propose a label augmentation method for equation set problems, using five normalization rules to generate equivalent labels. The method is applicable to any existing neural network models for equation set problems.
- We carefully compare the difference and performance improvements of these five rules. Experimental results show that the larger the difference with ground truth, the more significant the performance improvement.

2 Methodology

In this section, we give a detailed description of the five normalization rules and the model architecture.

2.1 Normalization Rules

An equation set problem (P, E) consists of a problem text P and an equation expression E. Different from previous studies on establishing models to learn the mapping $P \to E$, our goal is to learn the mapping $P \to (E, E^*)$, where E^* is generated by some normalization rules that can be summarized as follows (If E^* cannot be obtained from some normalization rules, $E^* = E$):

Rule 1: Equation swapping. This rule swaps the order of multiple equations. For example, we obtain $E^* : (2x + 4y = 50, x + y = 20)$ from $E : (x+y = 20, 2x+4y = 50)$, which shows that the order of equations does not affect the answer.

Rule 2: Variable substitution. This rule substitutes the unknown variables. For example, we obtain $E^* : (y+x = 20, 2y+4x = 50)$, which indicates that the naming of unknown variables does not affect the answer.

Rule 3: Commutative law. This rule swaps the left and right sides of an expression if the operators are '$*$' and '$+$'. For example, we obtain $E^* : (y + x = 20, y * 4 + x * 2 = 50)$, which shows that the '$*$' and '$+$' operators satisfy the commutative law.

Rule 4: New solution 1. This rule produces a new solution to the problem. Specifically, the expression of x is obtained from the first equation and substituted into the second equation; the expression of y is obtained from the second equation and substituted into the first equation. For example, we obtain the expression $x = 20 - y$ from the first equation $x + y = 20$. After substituting into the second equation, we obtain the new equation $2*(20-y)+4y = 50$. Similarly, we can get the new equation $x + (50 - 2x) \div 4 = 20$. Thus, we obtain $E^* : (x+(50-2x)\div 4 = 20, 2*(20-y)+4y = 50)$, which indicates that there is a new solution different from the ground truth.

Rule 5: New solution 2. This rule produces another new solution to the problem. We solve the equations directly to obtain the expressions for each unknown variable. For example, we obtain $E^* : (x = (4*20-50) \div (4-2), y = (50-2*20) \div (4-2))$, which shows that there is another new solution different from the ground truth.

As we can see, these rules produce an increasing difference with ground truth. Then an equation set problem (P, E) is transformed into (P, E, E^*).

2.2 Model Architecture

Our proposed label augmentation method is applicable to any existing neural network models. Since EPT model (Kim et al., 2020) has achieved the best performance on the equation set problems so far, we implement the label augmentation method on the model. EPT model also follows the encoder-decoder structure, where the encoder is the pre-trained language model Albert (Lan et al., 2020) and the decoder is the Transformer with the pointer network (Vinyals et al., 2015). Additionally, the equation E is transformed into a series of 3-tuples (f, a_1, a_2) in EPT model, where f is an operator and a_i is an operand. The decoder generates these 3-tuples in the sequence manner. Given an equation set problem (P, E, E^*), our model architecture is described as follows:

$$\boldsymbol{X} = \text{Encoder}(P) \tag{1}$$
$$S_1 = \text{Decoder}_1(\boldsymbol{X}) \tag{2}$$
$$S_2 = \text{Decoder}_2(\boldsymbol{X}) \tag{3}$$
$$\mathcal{L}_1 = -\log \Pr(E|S_1) \tag{4}$$
$$\mathcal{L}_2 = -\log \Pr(E^*|S_2) \tag{5}$$
$$\mathcal{L} = (\mathcal{L}_1 + \mathcal{L}_2)/2 \tag{6}$$

where $\mathcal{L}_i$ denotes the loss function of the i-th decoder. Pr is the cross-entropy loss function with the label smoothing approach, which sums up the loss of operators and operands in 3-tuples.

In the predicting stage, we perform beam searches for the two decoders respectively. Then we select the one with a higher score as the final result based on the beam score.

3 Datasets

We evaluate the performance of models on the English dataset **DRAW1K** and the Chinese dataset **HMWP**. DRAW1K (Upadhyay et al., 2016) contains 1,000 equation set problems, including 255 one-unknown-variable problems and 745 two-unknown-variable problems. HMWP (Qin et al., 2020) contains 5,470 problems, including 3,856 one-unknown-variable problems and 1,614 two-unknown-variable problems. For DRAW1K, we use the public training, development and test set. For HMWP, we split the training, development and test set at rate 3:1:1. The statistical information of datasets is shown in Table 2. At last, we employ the answer accuracy as the metric for measuring performance. In other words, the result is correct if the generated equations match the correct answer without considering the order of the answer.

Dataset	#Train	#Dev	#Test	Avg PL	Avg EL
DRAW1K	600	200	200	35.34	14.16
HMWP	3,282	1,094	1,094	61.99	13.21

Table 2: The statistical information of datasets, where Avg PL and Avg EL indicate the average problem length and average equation length respectively.

4 Experiments

4.1 Baselines

We compare our model with some state-of-the-art methods, including: **DNS** (Wang et al., 2017) directly generated equations with a sequence-based decoder, **SAU** (Qin et al., 2020) transformed equations into the tree structure and generated equations with a tree-based decoder, **DAG** (Cao et al., 2021) generated 3-tuples with a directed acyclic graph and **EPT** (Kim et al., 2020).

4.2 Implementation Details

The hyper-parameters we used are the same as EPT, including: 500 epochs, 2,048 batch sizes in terms of text or equation tokens, and 2 gradient accumulation. Since the learning rate and the warm-up epoch are decided by grid search, their values are listed in the Appendix. We use LAMB (You et al., 2019) optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. To facilitate the replication of experiments by researchers, we set the random seed to 1. At last, we perform the beam search with beam size 3. All experiments are implemented in pytorch 1.9.0 and run on Linux with NVIDIA Tesla V100.

Model	DRAW1K	HMWP	Model	DRAW1K	HMWP
DNS	36.8*	32.7*	EPT-B(2decoders) + 1	51.5	54.5
SAU	39.2*	43.7*	EPT-B(2decoders) + 2	55.0	**57.4**
DAG	44.4*	-	EPT-B(2decoders) + 3	54.5	57.3
EPT-B	**52.5**	**54.1**	EPT-B(2decoders) + 4	55.0	56.2
EPT-L	**57.5**	**58.5**	EPT-B(2decoders) + 5	**56.5**	56.3
EPT-B(1decoder) + 1	44.0	51.6	EPT-L(2decoders) + 4	**62.0**	60.3
EPT-B(1decoder) + 5	46.5	49.2	EPT-L(2decoders) + 5	60.5	**60.7**

Table 3: Results on DRAW1K and HMWP, where B and L represent the Albert-base and Albert-large models respectively. '*' denotes the accuracy via 5-fold cross-validation, which is reported by the paper (Lan et al., 2021). '1decoder' means that we use 1 decoder to train the ground truth and the equivalent labels produced by normalization rules. We rerun the EPT model, and the results may differ from the original paper.

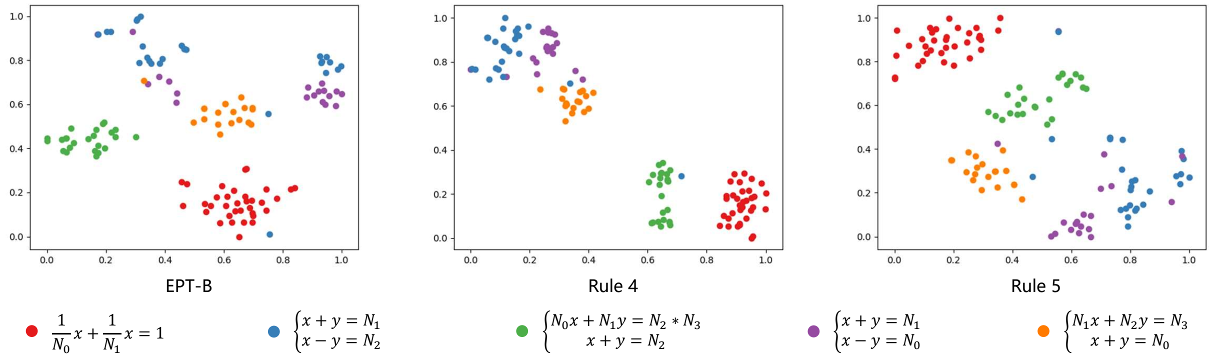

Figure 2: T-SNE visualization of the problem representation on DRAW1K.

4.3 Experimental Results

Table 3 depicts the results on two datasets, and we can observe the following phenomena:

(1) The effect of Rule 1 is barely improved and may be regressive. This is because Rule 1 is not applicable to one-unknown-variable problems and too simple for two-unknown-variable equations.

(2) Rule 2 and Rule 3 have similar improvements. For example, $x + y = 20$ is transformed into the same equation through Rule 2 and Rule 3, so Rule 2 and Rule 3 are similar in some cases.

(3) The improvements of Rule 4 and Rule 5 are obvious on DRAW1K, but they are not good on HMWP. This is because HMWP contains 900 one-unknown-variable nonlinear problems, which cannot be transformed by Rule 4 and Rule 5. This part of the data causes inconsistencies in the form of labels, resulting in performance degradation.

(4) The framework of two decoders is more suitable for the training of these equivalent labels. Since there are differences between the ground truth and augmented labels, using a single decoder confuses the training objective of the model. Even Rule 1 with the least difference has a significant performance degradation.

We obtain the problem representation by using the first-word embedding of problem text, and performing the T-SNE visualization (Rauber et al., 2016) in Figure 2. It can be seen that EPT-B does not distinguish these two parts well due to the similar forms of blue and purple ground truth. After training the equivalent labels produced by Rule 4 and Rule 5, there is some degree of distinction between the blue and purple labels, which indicates that the model can better establish the relationship between problem texts and equation expressions.

5 Conclusion

In this paper, we proposed a label augmentation method for equation set problems, summarizing 5 normalization rules to produce equivalent labels. To take advantage of these augmented labels, we replaced a single decoder with two decoders. Experimental results on both English and Chinese datasets show that our proposal has at most 4.5% improvement and better problem representation.

References

Yixuan Cao, Feng Hong, Hongwei Li, and Ping Luo. 2021. A bottom-up DAG structure extraction model for math word problems. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 39–46.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. *arXiv preprint abs/2109.13112*.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 3768–3779.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial examples for evaluating math word problem solvers. *arXiv preprint abs/2109.05925*.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2021. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. *arXiv preprint abs/2109.00799*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations*.

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 abs/2110.08464*.

Zhenwen Liang and Xiangliang Zhang. 2021. Solving math word problems with teacher supervision. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 3522–3528.

Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Reverse operation based data augmentation for solving math word problems. *arXiv preprint abs/2010.01556*.

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.

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*, 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*, pages 3780–3789.

Paulo E. Rauber, Alexandre X. Falcão, and Alexandru C. Telea. 2016. Visualizing time-dependent data using dynamic t-sne. In *18th Eurographics Conference on Visualization, EuroVis 2016 - Short Papers, Groningen, The Netherlands, June 6-10, 2016*, pages 73–77.

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. *arXiv preprint abs/2109.03034*.

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.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 2692–2700.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint abs/1910.03771*.

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*, pages 5299–5305.

Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. 2019. Reducing BERT pre-training time from 3 days to 76 minutes. *arXiv preprint abs/1904.00962*.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2020a. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2287–2305.

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

A Hyper-parameters

Model	Pre-trained Language Model
DRAW1K dataset	
EPT-B	Albert-base-v2
EPT-L	Albert-large-v2
HMWP dataset	
EPT-B	voidful/albert_chinese_base
EPT-L	voidful/albert_chinese_large

Table 4: The pre-trained language models used for EPT and our model.

Model	Learning Rate	Warm-up
DRAW1K dataset		
EPT-B	0.00176	12.5
EPT-L	0.00176	2.5
HMWP dataset		
EPT-B	0.0025	12.5
EPT-L	0.0025	2.5

Table 5: The best hyper-parameters for EPT and our model on DRAW1K and HMWP.

Table 4 shows the pre-trained language models used for EPT and our model, which can be downloaded from the Huggingface's `transformers` library (Wolf et al., 2019).

Table 5 depicts the best learning rate and warm-up epoch for EPT and our model.

B Math word problems

Arithmetic Word Problem	
Problem:	**Three** less than **4** times a number is **325**. What is the number?
Numbers:	3('Three'), 4, 325
Expression:	$(325 + 3) \div 4$
Answer:	82
Equation Set Problem	
Problem:	A number added to **6** is equal to **30** less than **four** times the number. What is the number?
Numbers:	6, 30, 4('four')
Equation:	$x + 6 = 4x - 30$
Answer:	12

Table 6: The examples of arithmetic word problems and equation set problems.

Arithmetic Word Problems			
Dataset	Problems	Avg PL	Avg EL
MAWPS	2,373	31.01	6.20
Math23K*	23,161	27.99	5.55
Ape210K*	210,488	43.05	7.74
Equation Set Problems			
Dataset	Problems	Avg PL	Avg EL
Alg514	514	41.40	13.08
DRAW1K	1,000	35.34	14.16
HMWP*	5,470	61.99	13.21

Table 7: The statistical information of MWP datasets, where '*' represents the Chinese dataset.

Math word problems can be divided into arithmetic word problems and equation set problems based on the form of the expressions. Table 6 and Table 7 show the examples and relevant datasets of arithmetic word problems and equation set problems. As shown in Table 7, the average problem length and equation length of equation set problems are usually longer than those of arithmetic word problems, which indicates that solving equation set problems is a more difficult task than arithmetic word problems.

C Data processing in EPT

In this section, we describe how EPT transforms an equation E into a series of 3-tuples and the vocabulary of the output.

Input position	Output index	3-tuple token f	a_1	a_2
0	-	`BEGIN`		
1	R_0	`VAR`		
2	R_1	`VAR`		
3	R_2	$+$	R_0	R_1
4	R_3	$=$	R_2	N_0
5	R_4	$*$	2	R_0
6	R_5	$*$	4	R_1
7	R_6	$+$	R_4	R_5
8	R_7	$=$	R_6	N_1
-	R_8	`END`		

Table 8: The equations $x + y = 20, 2x + 4y = 50$ processing in EPT, where N_i denotes the number values in the problem text.

Table 8 depicts the equation processing of the EPT, where `BEGIN` represents starting an equation, `VAR` represents generating a variable, and `END` represents ending an equation. R_i denotes the i-th

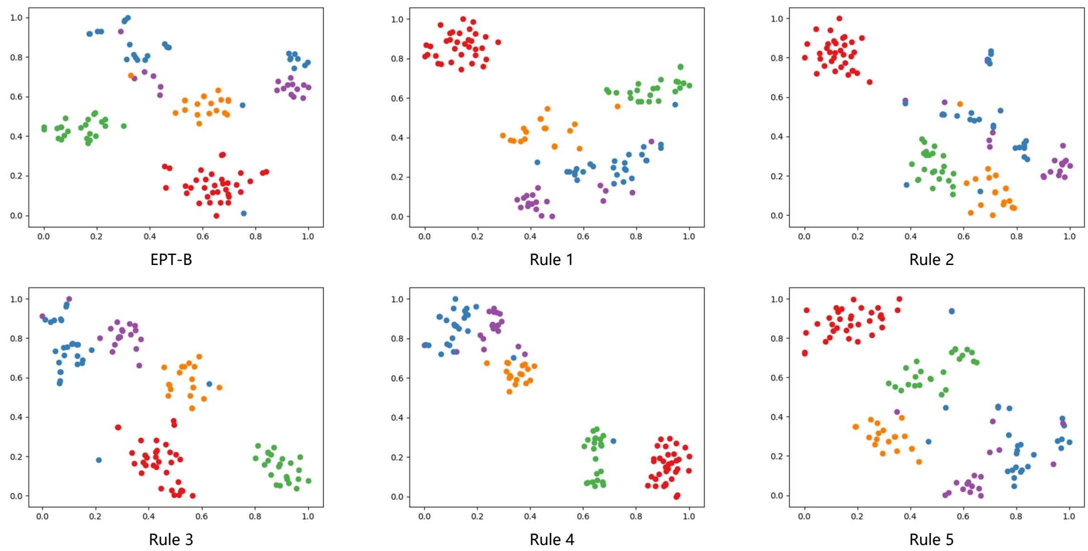

Figure 3: More results of T-SNE visualization on DRAW1K.

tuple and N_i indicates the i-th number value appearing in the problem text, e.g., $\{N_0 : 20, N_1 : 50\}$.

The output of EPT is a 3-tuple (f, a_1, a_2) at each step, where the vocabulary of the operator f is $\{+, -, *, \div, \wedge, =\}$, and the vocabulary of the operand a_i comes from: the numbers provided in the problem text (e.g., 20,50), constants (e.g., π, 1) and the prior 3-tuple (e.g., R_0).

D T-SNE visualization

Figure 3 shows more results of T-SNE visualization. Compared to problem representations learned by EPT, problem representations after training equivalent labels are more preferred to be clustered together if they have the same ground truth.

E Related Work

E.1 Equation set problems solving

As with solving arithmetic word problems, many researchers (Qin et al., 2020; Cao et al., 2021; Kim et al., 2020) still followed the encoder-decoder structure to generate equation expressions. Qin et al. (2020) proposed a new operator ';' to convert multiple expression trees into a general tree and generated equation(s) with a tree-based decoder. Some work (Cao et al., 2021; Kim et al., 2020) transformed the equations into a set of 3-tuples, generating these 3-tuples from bottom to top with a directed acyclic graph (Cao et al., 2021) or from left to right with a transformer decoder (Kim et al., 2020). Similar to the challenge of arithmetic word problems, these neural network models that use the ground truth as the only generation target do not have sufficient generalization performance.

E.2 Data augmentation/multi-task learning

To improve the generalization performance of the model, recent studies (Liu et al., 2020; Shen et al., 2021; Huang et al., 2021; Li et al., 2021; Qin et al., 2021) adopted data augmentation or multi-task learning framework to increase the difficulty of model training. Liu et al. (2020) swapped the descriptions of conditions and questions in the problem texts to obtain new arithmetic word problems. Liang and Zhang (2021) designed a teacher module to supervise the conformity between the problem representation and the ground truth. Shen et al. (2021) selected the correct expression with the highest ranking score from candidate expressions by a ranker module. Huang et al. (2021) first retrieved top-k similar problems for each problem, and then jointly trained the unsolved problems and each retrieved problem. Li et al. (2021) sought the most similar problem and easily confusing problem for each unsolved problem, enhancing the problem representation by contrastive learning. Qin et al. (2021) designed a series of auxiliary tasks, including number prediction task, commonsense constant prediction task, program consistency checker and duality exploiting task. Unlike these studies focused on designing auxiliary tasks on arithmetic word problems, we propose a label augmentation method for equation set problems, which is applicable to any existing neural network models for equation set problems.