Enhancing Seq2seq Math Word Problem Solver with Entity Information and Math Knowledge

Lei Li[1], Dongxiang Zhang[2], Chengyu Wang[3], Cheqing Jin[1], Ming Gao[1(✉)], and Aoying Zhou[1]

[1] School of Data Science and Engineering, East China Normal University, Shanghai, China
leili@stu.ecnu.edu.cn, {cqjin,mgao,ayzhou}@dase.ecnu.edu.cn
[2] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
zhangdongxiang@zju.edu.cn
[3] Alibaba Group, Hangzhou, China
chengyu.wcy@alibaba-inc.com

Abstract. Devising an automatic Math Word Problem (MWP) solver has emerged as an important task in recent years. Various applications such as online education and intelligent assistants are expecting better MWP solvers to process complex user queries that involve numerical reasoning. Current seq2seq MWP solvers encounter two critical challenges: ordinal indices without semantics and insufficient training data. In this work, we propose Entity Random Indexing to equip indices with semantics and design diverse representations of math expressions to augment training data. Experimental results show that our approach effectively enhances the seq2seq MWP solver, which outperforms strong baselines.

Keywords: Math word problem · Seq2seq model · Entity information · Math knowledge

1 Introduction

Math Word Problem (MWP) [20], which aims at answering a mathematical question automatically according to the textual content, is an essential language understanding task. A typical MWP is usually organized as sentences containing conditions and a final question about an unknown variable. Various domain-specific applications such as online education, e-commerce websites, and intelligent assistants, are expecting better MWP solvers to process complex queries involving numerical reasoning [3,8,12].

Table 1 shows an MWP example with its problem text, solution expression, and answer. Most solvers replace numeric values in problems with abstract indices since we are interested in modeling relations among operands rather than their specific values. It is a common practice for existing MWP solvers to

R. Chbeir et al. (Eds.): WISE 2022, LNCS 13724, pp. 370–385, 2022.
https://doi.org/10.1007/978-3-031-20891-1_26

Table 1. An example of MWP.

Original MWP
Problem: Mary has 2 books, Jack's books are 3 times over Mary's. Tom's books are 4 times over Jack's. Mike's books are 6 more than Tom's. How many times Mike's books are compared to Mary's books?
Solution: (2 * 3 * 4 + 6)/2 **Answer**: 15
Indexed MWP
Indices for Numbers:
{*NUM1*: 2, *NUM2*: 3, *NUM3*: 4, *NUM4*: 6}
Indexed Problem: Mary has *NUM1* books, Jack's books are *NUM2* times over Mary's. Tom's books are *NUM3* times over Jack's. Mike's books are *NUM4* more than Tom's. How many times Mike's books are compared to Mary's books?
Indexed Solution: (*NUM1***NUM2***NUM3*+*NUM4*)/*NUM1*
Indexed Answer: 15

index the numeric values according to their occurrence order and process them as regular tokens. For example, the token "NUM1" is used to refer to the first numeric value in the problem text.

Current seq2seq MWP solvers encounter two critical challenges. (1) To capture relations between operands rather than specific numbers, previous works replace numbers with ordinal indices. We suggest that such a way of handling numeric values can cause a drawback. Each frequent numeric token (e.g., "NUM1", "NUM2", etc.) is associated with various semantic contexts in the input problems; hence it is challenging for seq2seq models to learn effective representations that are able to disambiguate different semantic contexts. (2) Domain-specific MWP training data is not sufficient enough to train an MWP solver based on deep neural networks, which leads to model overfitting.

To overcome these two challenges, we enhance seq2seq MWP solvers with entity information and math knowledge. First, we should indexing numeric values according to related entities. Thus, a solver can infer semantics of indices of numeric values to better capture the information MWP. Second, we should utilize math knowledge to augment MWP training data since math knowledge enable us to produce a lot of variants for original math expressions of MWP.

Our contributions are summarized as follows: (1) We propose an algorithm named Entity Random Indexing which can equip indices with rich semantics. (2) We design diverse representations of math expressions to augment training data. (3) Based on the vanilla seq2seq model and our approaches (Entity Random Indexing and diverse representations of math expressions), we provide a new competitive MWP solver that outperforms strong baselines.

2 Related Work

2.1 Seq2seq-Based MWP Solvers

The earliest work to apply the seq2seq model was proposed by Wang et al. [17], where a seq2seq model is combined with a similarity-based retrieval model to generate expressions. Huang et al. [5] propose to incorporate copy and alignment mechanisms to seq2seq models for performance improvement. Wang et al. [16] observe that seq2seq models tend to suffer from equation duplication and propose an equation normalization method to normalize the duplicated equations.

2.2 Tree-Based MWP Solvers

Most state-of-the-art MWP solvers are based on trees since the solution can be naturally represented as an expression tree. GTS [18] develops a tree-structured neural network in a goal-driven manner to generate expression trees. Graph2Tree [21] combines the merits of the graph-based encoder and the tree-based decoder to generate better solution expressions. RODA [11] makes use of mathematical logic to produce new high-quality math problems and augments data for GTS to achieve better results.

2.3 Data Augmentation

Solving MWP is related to common tasks in NLP, such as Natural Language Inference (NLI) and Question Answering (QA), which can benefit from rule-based data augmentation. Kang et al. [6] propose NLI-specific logic-based data augmentation by replacing tokens or changing labels on the original training examples. Asai and Hajishirzi [1] propose a method that leverages logical and linguistic knowledge to augment labeled training data. Liu et al. [11] propose a reverse operation-based data augmentation method that makes use of mathematical logic to produce new high-quality math problems.

3 Methodology

In this section, we elaborate the techniques of the proposed approach for building MWP solvers.

3.1 Problem Statement

An MWP can be formulated as a triple $\{P, I, A\}$. The problem text P is a sequence of word tokens and numeric values. Let $T_p = \{t_1, \cdots, t_a\}$ denote the word tokens in P and $N_p = \{n_1, \cdots, n_b\}$ denote the set of numeric values in P. Thus, we have $P = \{p_1, \cdots, p_c | p_c \in T_p \cup N_p\}$. Given an MWP P, the goal is to map P to several computable intermediate sequences (CIS) which can be used to produce the answer of P. $I = \{i_1, \cdots, i_d\}$ is the set that consists

of CIS i_d. i_d only needs to be a parsable sequence for computation (e.g., a math expression, a postfix expression) and describes a computational process including numeric values N_p, the set of math operators $\{+, -, *, /\}$ and constants. Constants include some special values such as 3.14 (π), 1 and 2. Each correct i_d is able to produce the correct answer A to the problem text P. In this paper, we only consider problems whose the answers A are unique values. Furthermore, we convert fractional answers into decimals. Since manipulating floating-point numbers will result in rounding errors, we consider a problem to be "solved" when the absolute value of the difference between the produced and true answers is less than a small acceptable tolerance, i.e., 0.001.

Algorithm 1. Entity Random Indexing (ERI)

Input: $S_P = \{P_1, \cdots, P_h\}$, $P_h = \{P_{h,1}, \cdots, P_{h,c}\}$
S_P is original training data, P_h is one problem text, $P_{h,c}$ is one token in P_h
Output: D_t (the indexed training data)

```
Build S_E containing common entities and units.
Variables: Range and Occupied are hashtables; C_i is the times for indexing; C_j is
the length of the indexing range; C_n is the number of the indexing range.
repeat C_i times
    for each element E_g in S_E do
        Random(0, C_n) ∈ {0, 1, 2, ..., C_n − 1}
        Range[E_g] ← C_j * Random(0, C_n)
    Let S_{P,C} be a copy of S_P.
    for P_h in S_{P,C} do
        for P_{h,c} in P_h do
            if P_{h,c} is number then
                Find the entity or unit E_g that follows P_{h,c}.
                Start numbering r_s ← Range[E_g].
                while Occupied[r_s] = True do
                    r_s ← r_s + 1 // skip occupied indices.
                Occupied[r_s] ← True // occupy the index.
                Replace P_{h,c} with NUMr_s.
    Append S_{P,C} to D_t.
```

3.2 Entity Random Indexing

We provide a heuristic algorithm to handle numeric values based on an observation that most numeric values in MWP are followed by entities or units (e.g., "2 books" and "3 h"), which incorporate semantic contexts of the numeric values. Thus, we are motivated to allocate an index interval for each type of entity or unit. For example, the range NUM16-NUM20 is allocated to represent numeric values associated with "books". Furthermore, inspired by multi-headed self-attention [15], where multiple linear transformations are applied to the input

data for more effective representation learning, we randomly produce multiple indexing range divisions for the same entity or unit. We name our algorithm as Entity Random Indexing (ERI) and show the details in Algorithm 1. In contrast, the pointer-generator network [13] relies on massive data to learn copying tokens from the source to the target, while ERI is fully zero-shot and requires no training data.

3.3 Diverse Representations of Math Expressions

In this section, we offer a series of equivalent transformations based on math knowledge to augment the MWP training data. Given a solution expression, we propose four types of equivalent transformation, namely expression transformation, equation transformation, matrix transformation, and template transformation, which in total can generate 30 counterparts to the same answer. In the following, we assume that the solution expression in Table 1 is converted by ERI into

$$E : (NUM16 * NUM46 * NUM31 + NUM76)/NUM16$$

and use it to explain the idea of equivalent transformations. Note that transformed sequences may not be human-readable. However, they can be effectively calculable by the programs that we implement.

Expression Transformation. Our first equivalent transformation is to inject spaces to split operands and operators in E. We have:

$$E_1 : (NUM16 \ * \ NUM46 \ * \ NUM31 \ + \ NUM76)/NUM16.$$

The second alternative transformation is to apply postfix expression on E_1, i.e.,

$$E_3 : NUM16 \ NUM46 \ * \ NUM31 \ * \ NUM76 \ + \ NUM16/.$$

Note that E_3 can be converted uniquely back to E_1. Besides postfix expression, we can also use reversion as a transformation operator. When this operator is applied on E_1 and E_3, we obtain E_2 and E_4 respectively.

$$E_2 : NUM16/) \ NUM76 \ + \ NUM31 \ * \ NUM46 \ * \ NUM16 \ ($$

$$E_4 : /NUM16 \ + \ NUM76 \ * \ NUM31 \ * \ NUM46 \ NUM16$$

Equation Transformation. The idea of equation transformation is to introduce an additional unknown variable X and convert an arithmetic expression into its equivalent form of an equation. For example, expression E can be naturally converted into

$$E' : (NUM16 * NUM46 * NUM31 + NUM76)/NUM16 = X.$$

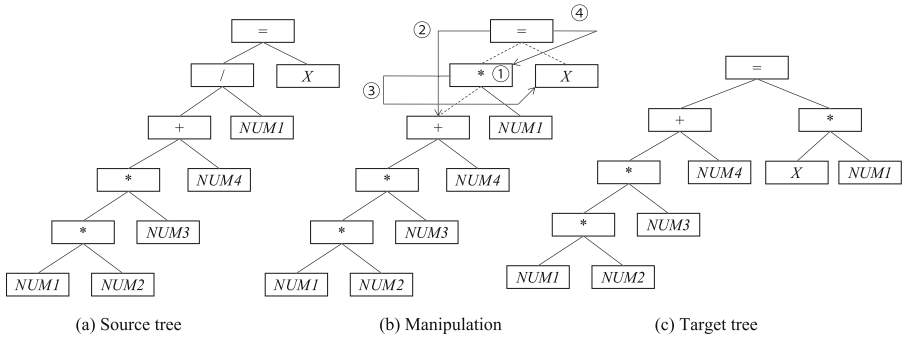

Fig. 1. Left shift of the equal sign based on tree manipulations.

From E', we can generate multiple equivalent variants of equations by shifting the equal sign leftwards. As shown in Fig. 1, we represent the E' as a binary tree. Left shift of equal sign involves four steps:

1. Replace root.left_child with reverse operator (e.g., / will be replaced by $*$).
2. Move root→left_child and root→left_child→right_child to root→right_child.
3. Put root→left_child→left_child as the new left child of root.
4. Put variable X as the left child of root→right_child

After the manipulation, we can generate the target tree that represents the target equation

$$E'' : NUM16 * NUM46 * NUM31 + NUM76 = X * NUM16.$$

Similarly, we can define the manipulation of the right shift to generate equivalent equations in Fig. 2.

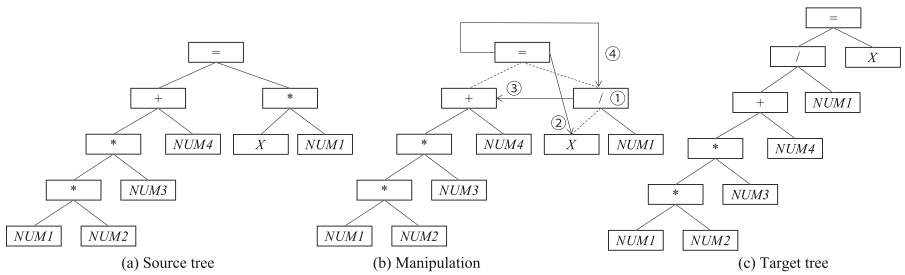

Fig. 2. Right shift of the equal sign based on tree manipulations.

As shown in Fig. 3, among the various equations generated by the left (or right) shift, we only select two of them based on the following criteria. First, we prefer the equation

$$E''' : NUM16 * NUM46 * NUM31 = X * NUM16 - NUM76$$

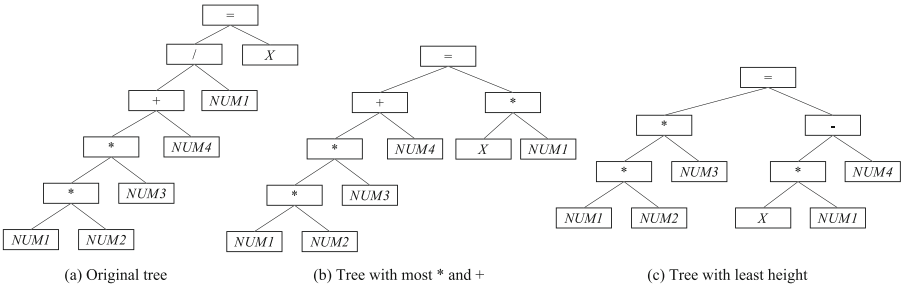

Fig. 3. Among the various equations generated by the left (or right) shift, we prefer equations with the minimal tree height or with most $*$ and $+$.

with the minimal tree height. Second, we prefer the equation

$$E'''' : NUM16 * NUM46 * NUM31 + NUM76 = X * NUM16$$

with most $*$ and $+$ for trying to reduce the operator that a solver need to learn from 4 types $(*, /, +, -)$ to 2 types $(*, +)$. Following the way producing equivalent transformations E_1 to E_4, we produce E_5 to E_8 based on E''' and list them in Table 2. For simplicity, we use N to replace NUM. Similarly, we produce E_9 to E_{12} based on E''''.

Table 2. Equation transformation.

E_5	$N16\ *\ N46\ *\ N31\ =\ X\ *\ N16\ -\ N76$
E_6	$N76\ -\ N16\ *\ X\ =\ N31\ *\ N46\ *\ N16$
E_7	$N16\ N46\ *\ N31\ *\ X\ N16\ *\ N76\ -\ =$
E_8	$=\ -\ N76\ *\ N16\ X\ *\ N31\ *\ N46\ N16$
E_9	$N16\ *\ N46\ *\ N31\ +\ N76\ =\ X\ *\ N16$
E_{10}	$N16\ *\ X\ =\ N76\ +\ N31\ *\ N46\ *\ N16$
E_{11}	$N16\ N46\ *\ N31\ *\ N76\ +\ X\ N16\ *\ =$
E_{12}	$=\ *\ N16\ X\ +\ N76\ *\ N31\ *\ N46\ N16$

Matrix Transformation. In this part, we propose a strategy that converts binary trees into sequences based on the adjacency matrix. Let m denote the number of operands in an expression and n denote the maximum number of frequencies for an operand to appear in an expression (since it is possible for the same operand to appear multiple times in an expression). We transform a binary tree with m operands into a square matrix M with the maximum size $(m * n) * (m * n)$. Each element $M[i, j]$ with $i \neq j$ stores a binary operator represented by an integer. We assign $\{1, 2, 3, 4\}$ for operators $\{+, -, *, /\}$ respectively. The

diagonal elements, i.e., $M[i, j]$ with $i = j$, store unary operators that allow us to build a smaller matrix to capture all the information in the binary tree. Details of unary operators are presented in Fig. 4, where we illustrate an example of matrix transformation.

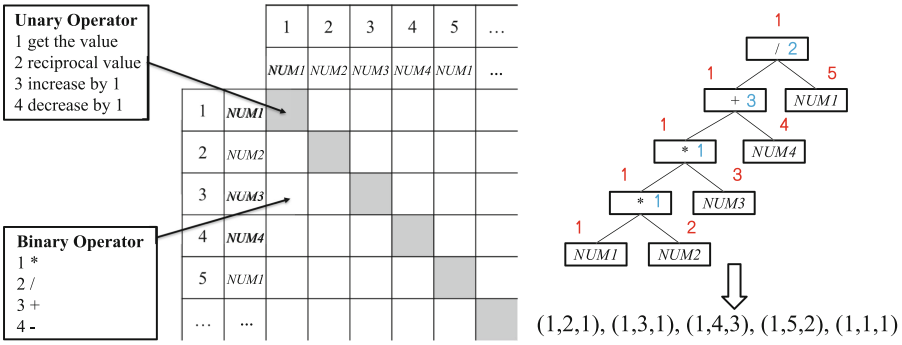

Fig. 4. Encoding a binary tree to a sequence of triples based on the adjacent matrix.

In Fig. 4, each row or column refers to an operand. Rows (and columns) "1, 2, 3, 4" refer to operands *NUM16*, *NUM46*, *NUM31*, *NUM76* respectively. Since *NUM16* appear twice in the expression, rows (and columns) "5, 6, 7, 8" also refer to indexes *NUM16*, *NUM46*, *NUM31*, *NUM76*. For matrix transformation, we perform post-order traversal to the binary tree. Each non-leaf node will output a triple. For example, in $(1, 2, 1)$, the first "1" stands for left operand *NUM16*, "2" stands for right operand *NUM46*, and the last "1" stands for binary operator $*$. It can be decoded as a computation "$NUM16 * NUM46$" and the temporary result will be stored at the first "1". Hence, its subsequent triple $(1, 3, 1)$ represents "$(NUM16 * NUM46) * NUM31$". We can perform recursive visit on the binary tree to generate the complete sequence of triples "$(1, 2, 1), (1, 3, 1), (1, 4, 3), (1, 5, 2), (1, 1, 1)$".

With the derived sequence "$(1, 2, 1), (1, 3, 1), (1, 4, 3), (1, 5, 2), (1, 1, 1)$", we further construct six equivalent variants to include more diversified knowledge and information. The equivalent transformation E_{13} is formatted as "1_2_1 1_3_1 1_4_3 1_5_2 1_1_1". 1) We reverse E_{13} to get E_{14} "1_1_1 1_5_2 1_4_3 1_3_1 1_2_1". 2) We swap the first and second elements of triples in E_{13} to get E_{15} "2_1_1 3_1_1 4_1_3 5_1_2 1_1_1". 3) We reverse E_{15} to get E_{16} "1_1_1 5_1_2 4_1_3 3_1_1 2_1_1". 4) We split triple tokens of E_{13} i.e. "1_2_1" into two sub tokens "1_2" and "1" to get E_{17} "1_2 1 1_3 1 1_4 3 1_5 2 1_1 1". 5) We reverse E_{17} to get E_{18} "1 1_1 2 1_5 3 1_4 1 1_3 1 1_2".

Readers can refer to Fig. 5 for the decoding process. For three numbers in a triple, the first and the second numbers represent a row. We retrieve its *NUM-index* and corresponding value and set the value to the corresponding position in the array. The third number refers to an operator. If the first number is not

equal to the second number, it is a binary operator. For a binary operator, we evaluate a temporary expression "first_number binary_operator second_number" i.e. *NUM16* $*$ *NUM46* and update the result to the position of its left operand in the array. For a unary operator, we employ the left operand as the argument and update the result to the position of the left operand in the array. Until all triples are processed, the final result is provided.

Template Transformation. Our template transformation is generated by replacing the numeric tokens with a specific placeholder (e.g., @) and consider the concrete indexes as arguments of the template. For example, the expression in E_1 "(*NUM16* * *NUM46* * *NUM31* + *NUM76*)/*NUM16*" can be encapsulated as a template "(@*@*@+@)/@" with sequence "*NUM16 NUM46 NUM31 NUM76 NUM16*" as the arguments. In this way, we can produce corresponding template transformations E_{19} to E_{30} for E_1 to E_{12}.

3.4 Architecture of Our MWP Solver

Based on ERI and equivalent transformations aforementioned, we present the architecture of our MWP solver named Solver with Entity and Math (SEM) in Fig. 6. Firstly, SEM uses ERI to preprocess problem texts and solutions in the MWP dataset. SEM repeatedly uses the indexed problem text as the source sequence and 30 equivalent transformations of MWP solutions as different target sequences to train 30 vanilla seq2seq models. After 30 seq2seq models are trained, SEM can solve an MWP by using its indexed problem text as the source sequence and let 30 seq2seq models produce 30 different target sequences. Then, SEM will recover numeric values by indices and compute each target sequence by the corresponding program. Invalid target sequences will be dropped out, while valid target sequences will result in numeric answers. A plurality vote [4,9,19] will be conducted on numeric answers and output the answer which receives the largest number of votes. Ties are broken arbitrarily.

4 Experiments

4.1 Experimental Configuration

Dataset. The dataset **Math23K** [17] is a large and widely-used benchmark[1] for MWP solvers. It contains 23,162 text problems annotated with solution expressions and answers. Each problem can be solved by one linear algebra expression with four types of operators $\{*, /, +, -\}$. The results reported in this paper are derived from 5-fold cross-validation.

Baselines. We compare our solver to a plenty number of baselines and state-of-the-art models. DNS [17] uses a vanilla seq2seq model to generate solution

[1] Zhao et al. [22] build a new large-scale and template-rich math word problem dataset named Ape210K. However, the publication of the dataset has been withdrawn in Arxiv (https://arxiv.org/abs/2009.11506).

Rows (cols) and indexes: {1: *NUM16*, *2*: *NUM46*, 3: *NUM31*, 4: *NUM76* , 5: *NUM16*, ...}
Indexes and values: {*NUM16*: 2, *NUM46*: 3, *NUM31*: 4, *NUM76*: 6}

	1	2	3	4	5	6
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	2	3				
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	6	3				
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	6	3	4			
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	24	3	4			
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	24	3	4	6		
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	30	3	4	6		
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	30	3	4	6	2	
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	15	3	4	6	2	
(1,2,1), (1,3,1), (1,4,3), (1,5,2), (1,1,1)	15	3	4	6	2	

Fig. 5. Calculating a sequence of triples to generate the answer of the MWP.

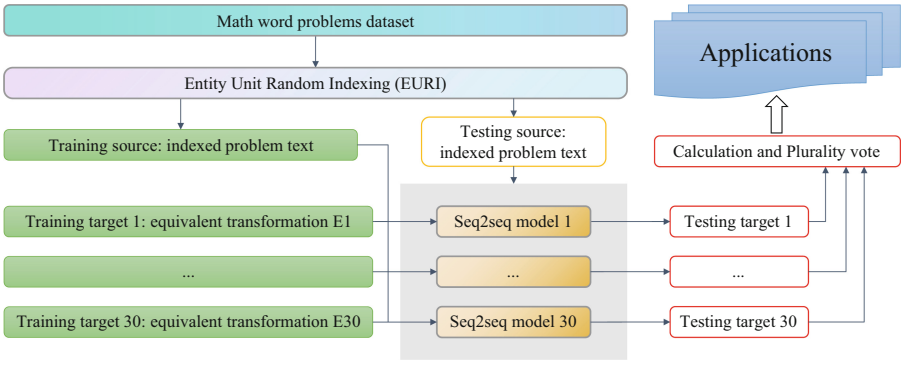

Fig. 6. Overall architecture of the SEM solver.

expressions. S-Aligned [2] designs the decoder with a stack to track the semantic meanings of operands. GROUP-ATT [10] adopts the idea of multi-head attention from the transformer [15]. GTS [18] develops a tree-structured neural network in a goal-driven manner to generate expression trees. Graph2Tree [21] combines the merits of the graph-based encoder and tree-based decoder to generate better solution expressions. RODA [11] makes use of mathematical logic to produce new high-quality math problems. ERNIE 3.0 [14] is a pre-trained language model with 10 billion parameters and is trained on a 4TB corpus consisting of plain texts and a large-scale knowledge graph.

Model Configuration. The vanilla seq2seq model used in our architecture is the 4-layer 16-head transformer [15], with $d_k = 12$, $d_v = 32$, and $d_{model} = 512$, where d_k and d_v are the dimension of keys and values respectively, and d_{model} is the output dimension of each sub-layer. We use the Adam optimizer [7] to train the model with the learning rate $1e^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and the dropout rate of 0.3.

4.2 Results

Overall Performance. Table 3 shows the accuracy of our SEM MWP solver and its competitors. We observe that SEM achieves competitive performance compared to state-of-the-art solvers. It proves that entity information and math knowledge can enhance seq2seq solvers to outperform both tree-based solvers and solvers relying on massive data pre-training.

Table 3. Answer Accuracy of SEM and baselines on **Math23K**.

Method	Answer accuracy (%)
DNS [17]	58.1
S-Aligned [2]	65.8
GROUP-ATT [10]	66.9
GTS [18]	74.3
ERNIE 3.0 [14]	75.0
Graph2Tree [21]	75.5
GTS+RODA [11]	76.0
SEM (Our Approach)	**77.1**

Effectiveness of ERI. To verify the effectiveness of ERI, we show frequency percentages of the ordinal indexing and ERI in Fig. 7. Using ordinal indexing, 46% numeric values are indexed as *NUM1*, 36% numeric values are indexed as *NUM2*, and 12% numeric values are indexed as *NUM3*. *NUM1*, *NUM2*, and *NUM3* are bound to uncertain semantics and are ambiguous to MWP solvers.

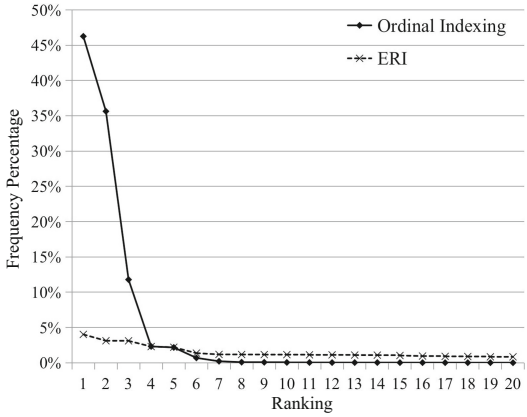

Fig. 7. Frequency percentages of the ordinal indexing and ERI.

However, when ERI is applied, indices are unambiguous since they are bound to certain entities. Furthermore, frequencies of ERI indices are similar and the most frequent indices do not exceed 5%. This is helpful for training a debiased neural network MWP solver.

Ablation Study. We conduct an ablation study to examine the effect of ERI and equivalent transformations, as shown in Table 4. When ERI is removed, there is a significant drop in accuracy (from 77.1% to 74.8%), implying that ERI is indeed effective. For the four types of solution transformations, we first investigate the accuracy when one of them is removed. We can see that matrix transformation is the most effective strategy. As we expected, diversifying the formats of output sequence representation can encourage seq2seq models to master math knowledge. We also conduct experiments to use only one type of solution augmentation, with interesting findings in Table 4. We can see that there is sharp performance degradation since the ensemble model works better if the underlying seq2seq models are more diversified. With only one type of augmentation available, the improvement by the ensemble model is rather limited. At last, the single vanilla seq2seq model that is trained by representation E_1 obtains 62.3% answer accuracy.

Correlation Analysis. As shown in Fig. 6 and Table 4, our SEM MWP solver obtains competitive performance based on plurality voting of solvers trained on diverse data.

Given t classifiers designed for binary classification, plurality voting can derive the correct prediction if at least $\lfloor t/2+1 \rfloor$ classifiers choose the correct class label. Assume that the outputs of the classifiers are independent, and each classifier has an accuracy p, implying that each classifier makes a correct classification at probability p. The probability of the ensemble for making a correct decision

Table 4. Ablation study of SEM.

Model configuration	Answer accuracy (%)
SEM (Full Implementation)	**77.1**
w/o. ERI	74.8, ↓ 2.3
w/o. Expression Transformation	76.8, ↓ 0.3
w/o. Equation Transformation	77.0, ↓ 0.1
w/o. Matrix Transformation	76.3, ↓ 0.8
w/o. Template Transformation	77.0, ↓ 0.1
w/ERI & Expression Transformation	65.3, ↓ 11.8
w/ERI & Equation Transformation	63.5, ↓ 13.6
w/ERI & Matrix Transformation	67.5, ↓ 9.6
w/ERI & Template Transformation	65.8, ↓ 11.3
Single seq2seq model trained on E_1	62.3, ↓ 14.8

Table 5. The Jaccard index between correct answers of two solvers.

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16
#1	1.00	0.74	0.72	0.74	0.74	0.73	0.74	0.74	0.73	0.73	0.65	0.73	0.74	0.74	0.74	0.74
#2	0.74	1.00	0.70	0.74	0.74	0.73	0.74	0.73	0.72	0.73	0.65	0.72	0.74	0.72	0.73	0.73
#3	0.72	0.70	1.00	0.70	0.69	0.69	0.70	0.70	0.69	0.69	**0.62**	0.69	0.71	0.70	0.70	0.70
#4	0.74	0.74	0.70	1.00	0.74	0.73	0.73	0.74	0.73	0.72	0.65	0.74	0.73	0.73	0.73	0.73
#5	0.74	0.74	0.69	0.74	1.00	0.74	0.75	0.74	0.74	0.73	0.65	0.73	0.73	0.73	0.73	0.73
#6	0.73	0.73	0.69	0.73	0.74	1.00	0.73	0.74	0.72	0.73	0.64	0.73	0.72	0.72	0.72	0.72
#7	0.74	0.74	0.70	0.73	0.75	0.73	1.00	0.74	0.73	0.72	0.66	0.73	0.73	0.73	0.73	0.73
#8	0.74	0.73	0.70	0.74	0.74	0.74	0.74	1.00	0.73	0.73	0.64	0.74	0.73	0.73	0.73	0.73
#9	0.73	0.72	0.69	0.73	0.74	0.72	0.73	0.73	1.00	0.74	0.66	0.75	0.72	0.72	0.72	0.72
#10	0.73	0.73	0.69	0.72	0.73	0.73	0.72	0.73	0.74	1.00	0.66	0.75	0.72	0.71	0.71	0.71
#11	0.65	0.65	**0.62**	0.65	0.65	0.64	0.66	0.64	0.66	0.66	1.00	0.66	0.64	0.64	0.64	0.65
#12	0.73	0.72	0.69	0.74	0.73	0.73	0.73	0.74	0.75	0.75	0.66	1.00	0.72	0.72	0.72	0.72
#13	0.74	0.74	0.71	0.73	0.73	0.72	0.73	0.73	0.72	0.72	0.64	0.72	1.00	0.77	**0.80**	0.78
#14	0.74	0.72	0.70	0.73	0.73	0.72	0.73	0.73	0.72	0.71	0.64	0.72	0.77	1.00	0.77	0.79
#15	0.74	0.73	0.70	0.73	0.73	0.72	0.73	0.73	0.72	0.71	0.64	0.72	**0.80**	0.77	1.00	0.78
#16	0.74	0.73	0.70	0.73	0.73	0.72	0.73	0.73	0.72	0.71	0.65	0.72	0.78	0.79	0.78	1.00

can be calculated using a binomial distribution. Specifically, the probability of obtaining at least $\lfloor t/2+1 \rfloor$ correct classifiers out of t is [4]:

$$P_{mv} = \sum_{k=\lfloor t/2+1 \rfloor}^{t} \binom{t}{k} p^k (1-p)^{t-k} \tag{1}$$

Lam and Suen [9] showed that 1) If $p > 0.5$, then P_{mv} is monotonically increasing in t, and $\lim_{t\to+\infty} P_{mv} = 1$ 2) If $p < 0.5$, then P_{mv} is monotonically decreasing in t, and $\lim_{t\to+\infty} P_{mv} = 0$. 3) If $p = 0.5$, then $P_{mv} = 0.5$ for any t.

Note that plurality voting fails to work well if the underlying models are highly correlated. Thus, our last experiment is to verify the correlation of our

solvers. We show the Jaccard index between correct answers of each pair of seq2seq solvers in Table 5 (Limited by space, we only show results between solver #1 to #16). The Jaccard index is computed as

$$J(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \tag{2}$$

where A_i represents the set of correct answers produced by i-th solver, A_j represents the set of correct answers produced by j-th solver. The $\#i$ solver is trained from equivalent representation E_i. The maximal Jaccard index 0.799 is achieved by the pair #13 and #15. The minimal Jaccard index 0.619 is achieved by the pair #3 and #11. E_{13} and E_{15} belong to the same type "Matrix Transformation" while E_3 and E_{11} belong to different types that are "Expression Transformation" and "Equation Transformation" respectively. Such evidence support that there is no highly correlated pair of solvers whose training data should be removed from the training dataset.

4.3 Demonstration

As shown in Fig. 8, we present a potential application of the SEM solver. Assume that we have an electric car with some attributes such as the maximum speed, the endurance mileage, the charging time, and the battery state. Our SEM solver can be trained to answer users' questions such as "I want to go to a place 700 km away, how many hours will it take as fast as possible?". The SEM solver will offer a human-readable math equation according to the answer that receives the most votes in our algorithm. Users can provide feedback on the answers. If "Accept" is clicked by a number of users, the question and the answer will be appended to the training set as the positive sample for iterative model improvement. If the "Reject" button is clicked, another candidate solution will be presented. Administrators of this system will randomly check user-generated data to filter out false positive samples. We are also working on adapting this demonstration to offer shopping recommendations according to the attributes of an item on e-commerce websites.

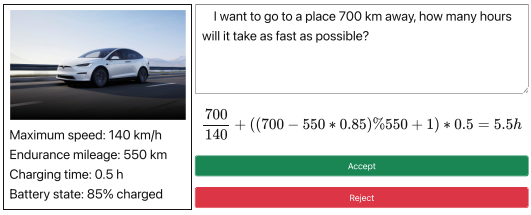

Fig. 8. A demonstration of the SEM solver.

5 Conclusion

In this paper, we propose Entity Random Indexing to equip indices with semantics and propose diverse representations of math expressions to augment training data. Experimental results show that our approaches enhance the seq2seq MWP solver to outperform strong baselines. As for future work, we seek to integrate the proposed approach with pre-trained language models to examine the generalization capability and to further improve the performance of MWP solvers. We are also planning to collect more and diverse MWP data for training and benchmarking. Accelerating the seq2seq model inference is also an interesting topic of making our solver to be a real-time application with low latency.

Acknowledgements. This work has been supported by the National Natural Science Foundation of China under Grant No. U1911203, the National Natural Science Foundation of China under Grant No. 61877018, and Alibaba Group through the Alibaba Innovation Research Program.

References

1. Asai, A., Hajishirzi, H.: Logic-guided data augmentation and regularization for consistent question answering. In: ACL 2020, pp. 5642–5650 (2020)
2. Chiang, T., Chen, Y.: Semantically-aligned equation generation for solving and reasoning math word problems. In: NAACL-HLT 2019, pp. 2656–2668 (2019)
3. Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., Gardner, M.: DROP: a reading comprehension benchmark requiring discrete reasoning over paragraphs. In: NAACL-HLT 2019, pp. 2368–2378 (2019)
4. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
5. Huang, D., Liu, J., Lin, C., Yin, J.: Neural math word problem solver with reinforcement learning. In: COLING 2018, pp. 213–223 (2018)
6. Kang, D., Khot, T., Sabharwal, A., Hovy, E.H.: Adventure: adversarial training for textual entailment with knowledge-guided examples. In: ACL 2018, pp. 2418–2428 (2018)
7. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR 2015 (2015)
8. Kirkland, P.K., McNeil, N.M.: Question design affects students' sense-making on mathematics word problems. Cogn. Sci. **45**(4) (2021)
9. Lam, L., Suen, C.Y.: Application of majority voting to pattern recognition: an analysis of its behavior and performance. IEEE Trans. Syst. Man Cybern. Part A **27**(5), 553–568 (1997)
10. Li, J., Wang, L., Zhang, J., Wang, Y., Dai, B.T., Zhang, D.: Modeling intra-relation in math word problems with different functional multi-head attentions. In: ACL 2019, pp. 6162–6167 (2019)
11. Liu, Q., Guan, W., Li, S., Cheng, F., Kawahara, D., Kurohashi, S.: RODA: reverse operation based data augmentation for solving math word problems. IEEE ACM Trans. Audio Speech Lang. Process. **30**, 1–11 (2022)
12. Nie, F., Wang, J., Yao, J., Pan, R., Lin, C.: Operation-guided neural networks for high fidelity data-to-text generation. In: EMNLP 2018, pp. 3879–3889 (2018)

13. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: ACL 2017, pp. 1073–1083 (2017)
14. Sun, Y., et al.: ERNIE 3.0: large-scale knowledge enhanced pre-training for language understanding and generation. CoRR abs/2107.02137 (2021)
15. Vaswani, A., et al.: Attention is all you need. In: NIPS 2017, pp. 5998–6008 (2017)
16. Wang, L., Wang, Y., Cai, D., Zhang, D., Liu, X.: Translating math word problem to expression tree. In: EMNLP 2018, pp. 1064–1069 (2018)
17. Wang, Y., Liu, X., Shi, S.: Deep neural solver for math word problems. In: EMNLP 2017, pp. 845–854 (2017)
18. Xie, Z., Sun, S.: A goal-driven tree-structured neural model for math word problems. In: IJCAI 2019, pp. 5299–5305 (2019)
19. Yue, K., Jin, R., Wong, C., Dai, H.: Federated learning via plurality vote. CoRR abs/2110.02998 (2021)
20. Zhang, D., Wang, L., Zhang, L., Dai, B.T., Shen, H.T.: The gap of semantic parsing: a survey on automatic math word problem solvers. IEEE Trans. Pattern Anal. Mach. Intell. **42**(9), 2287–2305 (2020)
21. Zhang, J., et al.: Graph-to-tree learning for solving math word problems. In: ACL 2020, pp. 3928–3937 (2020)
22. Zhao, W., Shang, M., Liu, Y., Wang, L., Liu, J.: Ape210k: a large-scale and template-rich dataset of math word problems. CoRR abs/2009.11506 (2020)