Automatic Math Word Problem Generation with Topic-Expression Co-attention Mechanism and Reinforcement Learning

Qinzhuo Wu, Qi Zhang*, Xuanjing Huang

***Abstract*—Taking several topic words and a math expression as input, the aim of math word problem generation is to generate a problem that can be answered by the given expression and related to these topic words. Considerable progress has been achieved by sequence-to-sequence neural network models in many natural language generation tasks, but these models do not effectively consider the characteristics of the math word problem generation task. They may generate problems that are unrelated to the topic words and expressions, and problems that cannot be solved. In this paper, we propose a new model, MWPGen, for automatically generating math word problems. MWPGen has a topic-expression co-attention mechanism to extract relevant information between topic words and expressions. Further, we fine-tune MWPGen with the solving result of the generated problem as the reward for reinforcement learning. MWPGen shows improved performance in popular automatic evaluation metrics and improves the solvability of generated problems.**

***Index Terms*—Math Word Problem Generation, Problem Generation, Natural Language Generation, Math Word Problem.**

Topic words: purchase, desk, chair
Math expression: N0 * (N1 + N2)
Ground truth: An elementary school bought N0 sets of desks and chairs. Each desk cost N1 and each chair cost N2. How much did this elementary school spend on desks and chairs?
Topic irrelevant problem: Each apple is worth N0. Alan bought N1 red apples and N2 green apples. How much did he spend? **Expression irrelevant problem:** An elementary school bought N0 sets of desks and chairs for N1. Each chair cost N2. How much did each desk cost? **Unsolvable problem:** An elementary school bought N0 sets of desks and chairs. Each desk cost N1 and each chair cost N2. How much did this elementary school spend on books?

Fig. 1. An example in which topic words and a math expression are needed to generate a corresponding math word problem. Several bad cases are shown that may be generated by the model.

I. Introduction

Math word problem generation is a novel task in natural language generation studies. Math word problems are widely used to facilitate education [1], [2] and in the development of large-scale datasets. In the elementary education stage, teachers need to construct or find a large number of math word problems for students' daily homework and regular tests. Automatic problem generation can assist teachers in their work and reduce their burden. Existing large-scale math word problem (MWP) datasets, like Math23K [3], are typically constructed by crawling web forums or expanding hand-crafted templates, which leads to high-cost annotation requirements or limited problem categories, respectively. The number of math word problems available from the web forums and hand-crafted templates is limited, but automatic problem generation can help people generate large-scale datasets, which expands the dataset capacity and reduces the annotation cost. Taking several topic words and a math expression as input, the task aims to generate a math word problem related to these topic words, and this problem can be solved to obtain the original math expression, as shown in Fig. 1. Without topic words, it is difficult for the model to automatically generate various math word problems using only math expressions. Here, we use topic words to help the model generate problems and post-verify the generated results. In recent years, neural network models have achieved good performance in many natural language generation tasks, such as machine translation [4], [5], reading comprehension [6], dialogue generation [7], [8], and image captioning [9], [10]. Zhou and Huang proposed a neural network model called MAGNET for generating math word problems, which encodes topic words and math expressions separately, and uses both topic and expression information to generate problems [11].

Although existing methods have achieved promising results, they do not effectively consider the characteristics of math word problem generation task. As this task must generate complete problems that are associated with given topic words and math expressions, the problems generated by these methods can face a number of issues: 1) The math word problems they generate may not be related to the given topic words and expressions. As shown in the examples in Fig. 1, the "topic irrelevant problem" does not consider the given topic words, and the "expression irrelevant problem" cannot be solved to obtain the original math expression. In this case, the model tends to generate general problems for different inputs. 2)

* Corresponding author.

The authors are with the School of Computer Science, Fudan University, Shanghai 200433, China (e-mail: qzwu17@fudan.edu.cn; qz@fudan.edu.cn; xjhuang@fudan.edu.cn).

The math word problems they generate may not be complete and solvable. When a key entity in the problem is wrong, the problem cannot be solved. As shown in the example in Fig. 1, the "unsolvable problem" asks for the price of "books" not mentioned in the previous text, which makes this problem unsolvable.

To address these issues, we propose the novel model MWPGen for automatically generating math word problems. First, we propose a topic-expression co-attention mechanism that extracts the correlated information between topic words and expressions, which enables the model to generate problems related to both inputs. We also convert the math expression into a pre-order traversal sequence of the expression tree and use adjacent node embeddings in the expression tree as additional embeddings. In this way, the model can capture structure and global semantic information of the math expression. Furthermore, we use a state-of-the-art math word problem solver to obtain math expression that corresponds to the generated problem, and determine whether this expression is the same as the original expression. To fine-tune our model, we use the results of the problem solver as rewards and apply reinforcement learning.

Our contributions can be summarized as follows:

- We propose a novel model for generating math word problems. The model has a topic-expression co-attention mechanism that can effectively extract correlated information between topic words and expressions. It also uses adjacent node embeddings in the expression tree as additional embeddings to capture the structure and global semantic information of the math expression.
- We use reinforcement learning to further fine-tune the model. We use a math word problem solving model to solve the generated problems, and use the results as rewards in reinforcement learning.
- We conducted experiments on a large-scale math word problem dataset and the results confirmed that the proposed model MWPGen outperformed baselines on popular automatic evaluation metrics. The results of the math word problem solving experiments also prove that the problems generated by MWPGen are more complete and solvable than those generated by other baseline models. In addition, human evaluation verified that these problems are more related to the given topic words and math expressions.

II. Models

In this section, we first describe the task of math word problem generation. Then, we introduce our proposed model MWPGen for generating math word problems. As shown in Fig. 2, the whole framework comprises a math word problem generator and a math word problem solver, the procedure for which is as follows: (1) The math word problem generator generates a problem for the given topic words and expression; (2) the generated problem is then sent to the solver to obtain its corresponding expression, and (3) the generated and original expressions are compared to produce a reward for reinforcement learning to fine-tune the MWPGen model.

A. Problem Definition

Given a topic word list $\mathrm{X^o} = \{x_1^o, x_2^o, \ldots, x_l^o\}$ and a math expression $\mathrm{X^e} = \{x_1^e, x_2^e, \ldots, x_m^e\}$, the generation task is to generate a math word problem $\mathrm{Y}= \{y_1, y_2, \ldots, y_n\}$, which is a sequence of n words related to the given topic words $\mathrm{X^o}$ that can be solved by the math expression $\mathrm{X^e}$. Here, l and m are the respective lengths of $\mathrm{X^o}$ and $\mathrm{X^e}$. Our goal is to estimate the probability distribution:

$$\mathbf{P}(\mathrm{Y}|\mathrm{X^o}, \mathrm{X^e}) = \prod_{\mathrm{t}=1}^{\mathrm{n}} \mathbf{P}(\mathrm{y_t}|\mathrm{y_{<t}}, \mathrm{X^o}, \mathrm{X^e}). \tag{1}$$

In this way, given a new input data pair $(\mathrm{X^o}, \mathrm{X^e})$, we can generate a new math word problem Y based on $\mathrm{P}(\mathrm{Y}|\mathrm{X^o}, \mathrm{X^e})$.

B. Math Word Problem Generator

Encoder:

Our model accepts two inputs, a topic word list $\mathrm{X^o}$ and a math expression $\mathrm{X^e}$, each of which is a sequence of words. We embed the words x_i^o, x_j^e in these two sequences as word embeddings $\mathbf{e}(x_i^o), \mathbf{e}(x_j^e)$ through a word embedding layer.

Additional embeddings for math expression:

The math expression can be converted into a binary tree structure with operators as internal nodes and numbers as leaf nodes. The goal of the model is to generate a pre-order traversal sequence of this expression tree. To better capture the structure and global semantic information of the math expression, we use the parent and child nodes of each word in the expression tree as additional embeddings, as shown in Fig. 3. We use this embedding strategy for two reasons. First, pre-ordered math expressions can better implicitly model tree structures than middle-ordered math expressions [12], [13]. Second, it can better capture long-distance dependencies. For example, in the math expression "(N0*N1)/(N2+N3)", the operator "/" directly depends on its child operators "*,+" instead of its nearby words "N1,), (, N2 ". Therefore, we use these adjacent node embeddings of the word in the expression tree as additional embeddings.

Then, we use two-layer bidirectional long short-term memory (BiLSTM) [14] to obtain the hidden states of each word in the math expression. The hidden states $\mathbf{h_i^e}$ are updated as follows:

$$\mathbf{h_i^e} = \mathrm{BiLSTM}(\mathbf{e}(x_i^e, x_{i,p}^e, x_{i,l}^e, x_{i,r}^e), \mathbf{h_{i-1}^e}), \tag{2}$$

where $\mathbf{e}(x_i^e, x_{i,p}^e, x_{i,l}^e, x_{i,r}^e)$ are the embeddings of the i-th word and its parent, left child, and right child in the expression tree, respectively, as shown in Fig. 3. We obtain the forward hidden states $\overrightarrow{\mathbf{h_i^e}}$ and backward hidden states $\overleftarrow{\mathbf{h_i^e}}$ by reading expression $\mathrm{X^e}$ in the forward and backward directions. We define the hidden states of the math expression $\mathbf{h_i^e}$ as the concatenation of the forward and backward hidden states, i.e., $\mathbf{h_i^e} = [\overrightarrow{\mathbf{h_i^e}} : \overleftarrow{\mathbf{h_i^e}}]$.

Topic-Expression Co-attention Mechanism:

We propose a topic-expression co-attention mechanism to generate a co-attention matrix M for the topic word embeddings $\mathbf{e}(x^o)$ and the expression hidden states $\mathbf{h^e}$:

$$\mathbf{M_{ij}} = \tanh(\mathbf{U}\ \mathbf{e}(x_i^o) \otimes \mathbf{h^{e_j}}\ ), \tag{3}$$

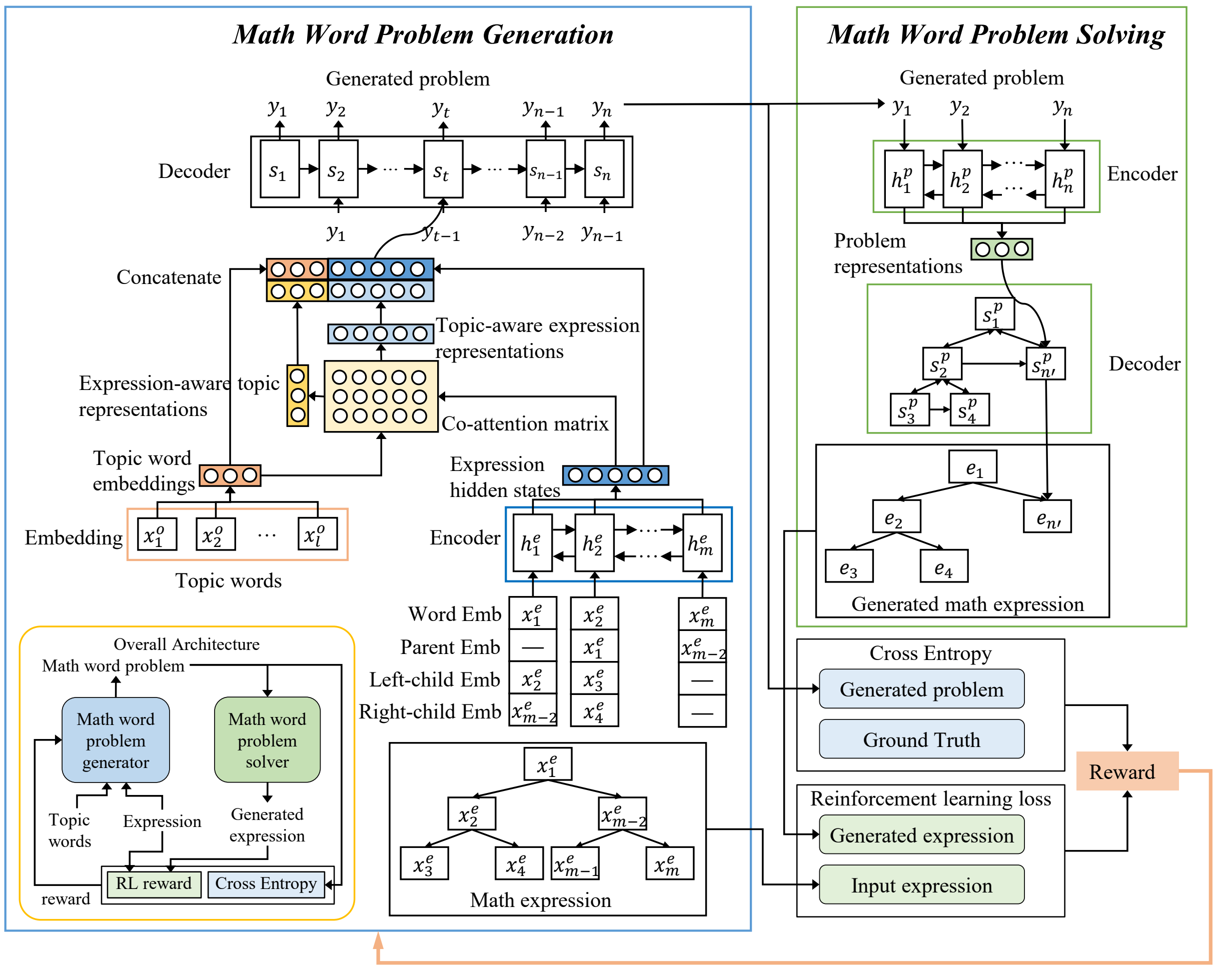
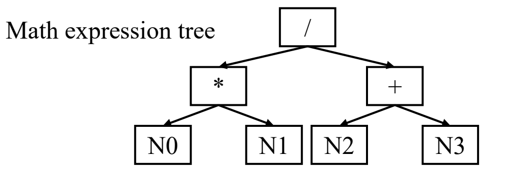


Fig. 2. Overview of our Math Word Problem Generation (MWPGen) network. (Left) Math word problem generation model. (Right) Math word problem solving model. The model is finally fine-tuned by rewards that are based on the similarity between the expression predicted by the MWP solving model and the original math expression.



Word	/	*	N0	N1	+	N2	N3
Parent	—	/	*	*	/	+	+
Left-child	*	N0	—	—	N2	—	—
Right-child	+	N1	—	—	N3	—	—

Fig. 3. An expression tree and the input embeddings of the math expression “(N0*N1)/(N2+N3)”. The input sequence is the pre-order traversal sequence of the expression tree. The input embeddings are a concatenation of the current word embeddings, parent embeddings, left-child embeddings, and right-child embeddings. “-” indicates that the current word has no parent or child nodes.

where $\otimes$ is a cross product function and $\mathbf{U} \in \mathrm{R}^{d\times d}$ is a weight matrix. In the co-attention matrix $\mathbf{M} \in \mathrm{R}^{l\times m\times d}$, l is the number of topic words, m is the length of the math expression, and d is the is the dimension of the hidden states.

We use this co-attention matrix $\mathbf{M}$ to compute topic-aware expression hidden states and expression-aware topic hidden states:

$$\begin{aligned}\alpha^o_{ij} &= \mathrm{softmax}(\mathrm{max\_pooling}(\mathrm{M})) \in \mathrm{R}^{l\times m},\\ \alpha^e_{ij} &= \mathrm{softmax}((\mathrm{max\_pooling}(\mathrm{M}))^{\mathrm{T}}) \in \mathrm{R}^{m\times l},\\ \mathbf{\hat{h}^o_i} &= \sum_{j=1}^{l} \alpha^o_{ij}\mathbf{e}(x^o_j),\ \ \mathbf{\hat{h}^e_i} = \sum_{j=1}^{m} \alpha^e_{ij}\mathbf{h^e_j}.\end{aligned} \tag{4}$$

Here, the $\mathrm{max\_pooling}(\cdot) : \mathrm{R}^{l\times m\times d} \rightarrow \mathrm{R}^{l\times m}$ runs on the last dimension of matrix M. α^o_{ij} and α^e_{ij} are co-attention weights distribution on the topic words and the math expression.

The final hidden states for the topic words and expression are as follows:

$$\mathbf{r^o_i} = [\mathbf{e}(x^o_i) : \mathbf{\hat{h}^o_i}], \quad \mathbf{r^e_i} = [\mathbf{h^e_i} : \mathbf{\hat{h}^e_i}]. \tag{5}$$

Here, [:] is the concatenation operation.

Decoder:

Given the last hidden states $\mathbf{h^e_m}$ of the expression as the initial hidden state $\mathbf{s_1}$ of the decoder, we use a one layer LSTM to generate the math problem. The hidden state $\mathbf{s_t}$ is updated as:

$$\mathbf{s_t} = \mathrm{LSTM}([\mathbf{e}(y_{t-1}), \mathbf{c_t}], \mathbf{s_{t-1}}). \tag{6}$$

Here, $\mathbf{s_{t-1}}$ and $\mathbf{e}(y_{t-1})$ are the decoder hidden state and the embedding of the generated word in the last time step. $\mathbf{c_t}$ is the context vector obtained by the attention mechanism [5]. Taking the topic word hidden states $\mathbf{r_i^o}$ and the expression hidden states $\mathbf{r_i^e}$ from the encoder, and last decoder hidden state $\mathbf{s_{t-1}}$, we can obtain $\mathbf{c_t}$ as:

$$\begin{aligned}\alpha_{ti} &= \mathrm{softmax}(\tanh(\mathrm{W_h}[\mathbf{r_i^o} : \mathbf{r_i^e}] + \mathrm{W_s}\mathbf{s_{t-1}})),\\ \mathbf{c_t} &= \sum_{i=1}^{l+m} \alpha_{ti}[\mathbf{r_i^o} : \mathbf{r_i^e}],\end{aligned} \tag{7}$$

where $\mathrm{W_h}$ and $\mathrm{W_s}$ are the weight matrices. α_{ti} denotes the attention distribution on the hidden states $\mathbf{r^o}$ and $\mathbf{r^e}$ at time step t.

Finally, using the context vector $\mathbf{c_t}$ and hidden state $\mathbf{s_t}$, the probability distribution of generating y_t is calculated as:

$$\begin{aligned}\mathbf{P}(y_t|y_{<t}, \mathrm{X^o}, \mathrm{X^e}) &= \mathrm{softmax}(\mathrm{W_g}[\mathbf{s_t} : \mathbf{c_t}]),\\ L_{loss} &= -\sum\nolimits_{t=1}^{n} \log \mathbf{P}(y_t|y_{<t}, \mathrm{X^o}, \mathrm{X^e}).\end{aligned} \tag{8}$$

During training, we optimized the probability of generating y_t with a cross-entropy loss function. During the test, we used beam search to generate the problem. We set the beam size to 5. At the first time step, we selected the top 5 words with the highest probability under the current distribution as the first word of the 5 candidate output sequences. Subsequently, at each time step, based on the candidate output sequences of the last time step, we selected the top 5 words with the highest probability under the current distribution. Finally, the sequence with the highest probability was selected from the 5 candidate sequences as the generated problem.

C. Math Word Problem Solver

After generating a math word problem Y= $\{y_1, y_2, \ldots, y_n\}$, Y is sent to a pre-trained math word problem solver (MWP solver) to obtain its math expression. The structure of this MWP solver is shown below.

Encoder:

The MWP solver takes the generated problem Y as input, encodes this sequence, and then passes it to a two-layer bidirectional LSTM:

$$\mathbf{h_i^p} = \mathrm{BiLSTM}(\mathbf{e}(y_i), \mathbf{h_{i-1}^p}). \tag{9}$$

Decoder:

Following the method proposed by [13], we use a tree-structured gated recurrent unit (GRU) [15] decoder with an attention mechanism to generate math expressions in a pre-order traversal from top to bottom:

$$\begin{aligned}\mathbf{s_t^p} &= \mathrm{GRU}(\mathbf{e}(e_{t-1}, e_{t,p}, e_{t,l}), \mathbf{c_t^p}, \mathbf{s_{t-1}^p})\\ \alpha_{ti}^p &= \mathrm{softmax}(\mathrm{W_s}\mathbf{s_{t-1}^p} + \mathrm{W_h}\mathbf{h_i^p})\\ \mathbf{c_t^p} &= \sum\nolimits_{i=1}^{n} \alpha_{ti}^p \mathbf{h_i^p}.\end{aligned} \tag{10}$$

At time step 1, we use the last hidden states $\mathbf{h_n^p}$ of the generated problem to initialize the decoder state $\mathbf{s_1^p}$. Here, e_{t-1},$e_{t,p}$ and $e_{t,l}$ represent the output words of the last node, parent node, and left sibling node of the current node e_t, respectively. If the current node does not have a parent node or left sibling node in the generated expression tree, we pad it with a PAD token.

In addition, we introduce a copying mechanism [16] to enable the model to either generate a word e_t from the vocabulary V or copy a word from the input problem Y. At time step t, based on the context vector $\mathbf{c_t^p}$ and the decoder state $\mathbf{s_t^p}$, this mechanism calculates a copy gate value $g_p \in [0, 1]$ to determine whether the word e_t is generated or copied:

$$\begin{aligned}g_p &= \sigma(\mathrm{W_s}\mathbf{s_t^p} + \mathrm{W_c}\mathbf{c_t^p}),\\ \mathbf{P_c}(e_t) &= \sum\nolimits_{e_t=\mathrm{y_i}} \alpha_{ti}^p,\\ \mathbf{P_g}(e_t) &= \mathrm{softmax}([\mathrm{W_g}\mathbf{s_t^p} : \mathbf{c_t^p}]),\\ \mathbf{P}(e_t|e_{<t},\mathrm{Y}) &= g_p\mathbf{P_c}(e_t) + (1-g_p)\mathbf{P_g}(e_t),\end{aligned} \tag{11}$$

where $\mathrm{W_s}$, $\mathrm{W_c}$ and $\mathrm{W_g}$ are weight matrices and σ is a sigmoid function. We obtain the final probability distribution $\mathbf{P}(e_t|e_{<t},\mathrm{Y})$ over both the generate distribution $\mathbf{P_g}(e_t)$ and copy distribution $\mathbf{P_c}(e_t)$.

During the test, at each time step, if e_t is an operator, this means that the current node is an internal node, and the decoder continues to generate its left child nodes. If e_t is a number, it is a leaf node, so the decoder will generate right child nodes for the previous internal nodes. Once the children of all the internal nodes have been generated, the generated expression sequence E= $\{e_1, e_2, \ldots, e_{n'}\}$ can be transformed into a complete tree and the decoding process is terminated.

D. Reinforcement learning

In each training iteration, we first use MWPGen to generate the math word problem Y based on the topic words $\mathrm{X^o}$ and expression $\mathrm{X^e}$, and then use the MWP solver to generate the math expression E based on Y. We use reinforcement learning [17] to fine-tune our model.

To do so, we define $r(y) = \exp(\mathrm{E}, \mathrm{X^e}) + \mathrm{ans}(\mathrm{E}, \mathrm{X^e})$ as a reward function for the generated problem Y, which is obtained by checking the expression correctness and the answer correctness between the generated expression E and the original expression $\mathrm{X^e}$. If these two expressions are the same, the $\exp(\mathrm{E}, \mathrm{X^e})$ is set to 1, otherwise it is 0. If these two expressions can be executed to produce the same answer, the $\mathrm{ans}(\mathrm{E}, \mathrm{X^e})$ is set to 1, otherwise it is 0. We also consider the correctness of the answer because the model may generate a correct expression but differ from the original expression, e.g., if the generated expression is "4+5" but the original expression is "5+4".

The loss function for this reinforcement learning is defined as:

$$L_{RL} = -(r(y^s) - r(y^*)) \sum \log \mathbf{P}(e_t|e_{<t},\mathrm{Y}). \tag{12}$$

$r(y^s)$ is the reward for the generated problem Y and $r(y^*)$ is a baseline reward to reduce the variance. Like the self-critical sequence training (SCST) strategy [18], y^* is estimated by using the greedy search results of the MWPGen model.

TABLE I
STATISTICAL ANALYSIS OF THE MATH23K DATASETS. l_o, l_e, l_p ARE THE RESPECTIVE AVERAGE LENGTHS OF THE TOPIC WORDS, EXPRESSION, AND PROBLEM.

	Num	l_o	l_e	l_p
Train	18,530	3.74	5.54	29.38
Test	4,652	3.74	5.61	29.82

E. Training

To jointly train the generation model with the above two loss functions, we minimize the total loss during the training:

$$L_{total} = (1-\beta)L_{loss} + \beta L_{RL} \tag{13}$$

where β controls the magnitude of the reinforcement loss. By minimizing the above loss function, our model can be expected to generate complete and solvable math word problems.

III. EXPERIMENT

A. Dataset

We evaluated our model on the Math23K [3] dataset, which contains 23,196 math word problems along with matching math expressions and answers. We filtered from this dataset any expressions that could not be executed to generate an answer, which reduced the size of the dataset to 23,182.

For the math word problem generation task, each problem-expression pair requires a corresponding topic word list. To simulate the real-world situation in which teachers create math word problems, for each problem, we provided a topic word for the problem domain and several topic words for the entities in the problem. The methods we used to select topic words are as follows:

Topic words for the problem domain:

We divided this dataset into 20 domains, e.g., “Engineering, Geometry, Purchase, Fraction”. For each problem, we simply checked whether this problem contains the keywords of each domain, as described in [12]. If the problem contained none of the domain keywords, we set the domain topic word of the problem to “Other”.

Topic words for the problem entities:

We detected entities in the problem using Hownet [19], i.e., a knowledge graph about Chinese words and concepts. We selected the nouns in Hownet as our entity vocabulary. If none entity in the vocabulary was detected in the problem, we did not provide any entity topic word for this problem. If there are more than four entities were detected in the problem, we randomly selected four of them as entity topic words.

Finally, we provided one domain topic word and from zero to four entity topic words for each problem, with the average length of the topic word list being 3.74. Here, we use topic words and math expressions as input, and math word problems as output. For evaluation, we randomly split the dataset into training and test sets at a 80:20 ratio. Table I shows statistical details of the dataset.

B. Implementation Details

We used Pytorch for our implementation[1]. We choose words that appear more than 5 times in the training set or appear as topic words to build a vocabulary V, and replace words that are not in the vocabulary with a UNK token. The size of vocabulary is 3,550. The word embedding dimension is 128 and the hidden dimension of the encoder and decoder is 256. The batch size is 64, and the learning rate of the Adam optimizer [20] is 0.001. We set β in our loss function to 0.5, and dropout rate [21] is 0.5.

During training, first, we trained 200 epochs of the math word problem solver described in Section II-C. Then, we trained 200 epochs of the proposed MWPGen model. During decoding, we used a beam search with a beam size of 5. We used the same parameter settings to train the baselines and our MWPGen.

C. Baselines

Here, we compare the performance of our model with the following baselines.

- **Seq2Seq [5]** is based on a sequence-to-sequence (seq2seq) model with attention mechanism. Seq2Seq consists of a two-layer bidirectional LSTM encoder and a LSTM decoder with attention mechanism. We send topic words and math expressions to the encoder in order, and the topic words and expressions are separated by a EOS token. In fact, the baseline can be seen as a ablation model of MWPGen without the adjacent node embeddings in the expression tree, the topic-expression co-attention mechanism and reinforcement learning. The only difference left is that MWPGen encodes the expression and topic words separately.
- **ConvS2S [22]** is based on a convolutional seq2seq network. It has the same input, decoder and attention mechanism as the Seq2Seq baseline. In the convolutional layer, it has filters of length 1, 3, 5, and 7. All hyper-parameter settings are the same as for the MWPGen.
- **NQG++ [23]** is a seq2seq model with copy and attention mechanisms. We use the same topic words and math expression input as the Seq2Seq baseline. To use it in the math word problem generation task, we removed features about the answer position and lexical information.
- **MAGNET [11]** is a GRU-based seq2seq model with a maxout layer and entity-enforced loss. It encodes topic words and math expressions separately, and uses both topic and expression information to generate math word problems. They use entity forced loss to ensure that entities in the input appear in the generated math problems. We use a two-layer bidirectional LSTM in place of a GRU encoder for a fairer comparison with our model.

D. Results

We employed the automatic metrics BLEU (1-4) [24], ROUGE-L [25], and CIDEr [26] to evaluate the n-gram similarity between the generated problems and the ground

[1]https://pytorch.org/

TABLE II
EVALUATION RESULTS ON AUTOMATIC METRICS. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	BLEU1	BLEU2	BLEU3	BLEU4	Rouge_L	CIDEr
Seq2Seq [4]	0.5554	0.4303	0.3498	0.2914	0.5498	2.2446
ConvS2S [22]	0.5395	0.4147	0.3347	0.2767	0.5384	2.1028
NQG++ [23]	0.5511	0.4260	0.3459	0.2881	0.5466	2.2190
MAGNET [11]	**0.5676**	0.4383	0.3544	0.2937	0.5524	2.2596
MWPGen	0.5658	**0.4448**	**0.3638**	**0.3037**	**0.5659**	**2.3166**

TABLE III
SOLVABILITY OF THE GENERATED MATH WORD PROBLEMS. EVALUATED BY A STATE-OF-ART MATH WORD PROBLEM SOLVING MODEL GTS [13]. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	Expression	Answer
Ground Truth	63.43%	75.75%
Seq2Seq [4]	51.42%	60.25%
ConvS2S [22]	43.54%	52.14%
NQG++ [23]	47.06%	55.59%
MAGNET [11]	55.57%	62.85%
MWPGen	**57.71%**	**66.43%**

truths. These automatic evaluation metrics reflect the fluency of the generated problem. However, they have limited abilities to reflect the solvability and completeness of the generated problem. For example, for the ground truth "each apple costs me N1", here are two sentences: Sentence A "I spent N1 on each apple" and Sentence B "each desk costs me N1". Sentence A has the same meaning as the ground truth, but it is not as close to the ground truth as Sentence B, which may result in a lower evaluation score.

To further evaluate the quality of the generated problems, we used a pre-trained math word problem solving model to solve the generated problems and checked whether the results obtained were the same as the original math expressions and answers. In this study, we used the state-of-the-art model GTS released by [13] for math word problem solving. It is a sequence-to-tree model that uses math word problems as input and generates expression trees from top to bottom.

The evaluation metrics include expression accuracy and answer accuracy, indicating whether these mathematical problems can be solved to obtain the original expression and the original answer. For example, the generated expression is "4+5" and the original expression is "5+4", which indicates that solving this problem has obtained a wrong expression and a correct answer. Therefore, the answer accuracy is always higher than the expression accuracy because sometimes the model generates a correct result with an expression different from the original one.

Table II shows the automatic metric evaluation results of our model compared with other baselines. Using problems generated by the different baselines as inputs, Table III shows the accuracies of the expressions and answers to these problem. We have the following observations:

1) Seq2Seq model performed slightly better than the NQG++ on automatic metrics, and its solvability is better than NQG++. We believe this is because the copy mechanism of NQG++ will copy keywords directly from inputs during the generation process. This mechanism may reduce the fluency and solvability of the generated problems.
2) MAGNET performed better than other baselines on automatic metrics and solvability. MAGNET uses entity-enforced loss to ensure that the entities in the generated math problem are highly relevant to the words in the given input. Although this additional loss may reduce the fluency and completeness of the generated problem, it is helpful for generating topic-related and expression-related problems.
3) Our proposed model MWPGen achieves competitive results compared with baselines on the automatic metrics. In addition, the solvability of the problems generated by MWPGen is better than these baselines. The observations in the next section also confirm the fluency and completeness of the problems generated by the proposed model. We attribute these improvements to the combination of all three components in MWPGen.

TABLE IV
HUMAN EVALUATION RESULTS. THESE METRICS ARE RATED ON A 1-3 SCALE (3 FOR THE BEST). THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	Fluency	Completeness	Expression Accuracy	Topic Words Relevance
Seq2Seq [4]	2.49	2.41	1.98	2.51
ConvS2S [22]	2.37	2.49	1.98	2.33
NQG++ [23]	2.45	2.50	1.94	2.39
MAGNET [11]	2.19	2.27	2.01	**2.69**
MWPGen	**2.62**	**2.54**	**2.18**	2.51

E. *Human Evaluation*

In addition, we can see that automatic evaluation metrics and problem solving model accuracies are not always related to human judgments on the correctness of a math word problem, human evaluation can help us to better evaluate its quality. We conducted human evaluation comparing generation problems from the baselines mentioned above and our model. Specifically, we consider four metrics in human evaluation:

- **Fluency** measures whether a problem is grammatically correct and is fluent to read.
- **Completeness** measures whether a problem has a clear question clause, and provides enough information in the description part to solve the question.
- **Expression Accuracy** measures whether a problem can be solved to obtain the given math expression.
- **Topic Words Relevance** measures whether a problem is relevant to all given topic words.

For human evaluation, we used the baselines mentioned above to compare with our model. We randomly selected 100 pairs of topic word lists and math expressions from the test set, and asked 3 native speakers to evaluate the generated problems of each model. For each metrics, we ask the reviewer to rate the problems on a 1-3 scale (3 for the best).

TABLE V
ABLATION STUDY OF THE ADJACENT NODE EMBEDDINGS IN THE EXPRESSION TREE. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	BLEU1	BLEU2	BLEU3	BLEU4	Rouge_L	CIDEr	Expression	Answer
MWPGen-base (seq2seq)	0.5654	0.4440	0.3632	0.3032	0.5657	2.3100	50.52%	60.75%
MWPGen-base with GCN	0.5647	0.4442	0.3638	0.3040	0.5646	2.3247	49.27%	59.78%
MWPGen-base with TreeLSTM	0.5375	0.4133	0.3332	0.2751	0.5378	2.1175	38.82%	46.37%
MWPGen-base with AdjEmb	**0.5663**	**0.4459**	**0.3654**	**0.3054**	**0.5677**	**2.3392**	**53.50%**	**63.06%**
MWPGen without AdjEmb	0.5643	0.4422	0.3609	0.3007	0.5649	2.3046	56.00%	65.05%
MWPGen without AdjEmb, with GCN	0.5658	0.4439	0.3625	0.3022	0.5656	**2.3201**	56.26%	64.57%
MWPGen without AdjEmb, with TreeLSTM	0.5420	0.4181	0.3379	0.2797	0.5434	2.1409	45.85%	52.37%
MWPGen	**0.5658**	**0.4448**	**0.3638**	**0.3037**	**0.5659**	2.3166	**57.71%**	**66.43%**

Results of each human evaluation metric are presented in Table IV. We can see that:

1) As for Topic Words Relevance, MAGNET gets the best score. MAGNET also achieved competitive results in Expression Accuracy. But for Fluency and Completeness, MAGNET does not perform as well as other baselines. The reason for this may be that its entity-enforced loss module force MAGNET to generate solvable problems containing words from the input, but makes the generated problems unnatural or incomplete.
2) For Fluency and Topic Words Relevance, ConvS2S does not perform as well as other baselines. Human evaluation found that compared with other baselines, ConvS2S generated simpler and shorter problems. These problems are usually not related to the topic words. Therefore, even if ConvS2S is competitive on Completeness and Expression Accuracy, we still believe that ConvS2S does not perform as well as other baselines.
3) MWPGen gets the best or competitive performance in each metric. It achieves the best performance on Fluency, Completeness and Expression Accuracy. We attribute the superior performance of MWPGen to two properties: MWPGen uses adjacent node embeddings in the expression tree, and thus better captures the structure and global semantic information of the math expression. MWPGen uses the quality and solvability of the generated problems as rewards to fine-tune the model, which can improve the score of expression accuracy. In this way, these two properties improve the expression accuracy of MWPGen. In addition, the rewards of reinforcement learning promote that the generated problems can be processed by the MWP solving model, which also greatly improves the fluency of MWPGen and slightly improves the completeness.

F. Ablation study

Effect of adjacent node embeddings in the expression tree: To verify the effectiveness of the adjacent node embeddings in the expression tree, we first conduct an ablation study on the adjacent node embeddings. Table V shows evaluation results for several variants of our proposed model on Math23K dataset. The definitions of the models under comparison are:

- **MWPGen-base:** It is a ablation model of MWPGen without the adjacent node embeddings in the expression tree, the topic-expression co-attention mechanism and reinforcement learning. It can be seen as a basic sequence to sequence model.
- **MWPGen-base with AdjEmb:** It adds adjacent node embeddings in the expression tree to the “MWPGen-base” for better comparison.
- **MWPGen without AdjEmb:** It is a ablation model of MWPGen which remove the adjacent node embeddings of the current word in the expression tree.
- **MWPGen:** It is the complete version of our proposed model with all three components.

In addition, we explored two other popular approaches to capture the structure and global semantic information of the math expression, as follows:

- **Graph Convolutional Network (GCN) [27]:** We use a two-layer graph convolutional network in place of the adjacent node embeddings for a fairer comparison with our model. Math expressions are converted into expression trees, where operators are internal nodes and numbers are leaf nodes. We use GCN to update the node state by aggregating its neighbor nodes in the expression tree.
- **Tree-LSTM [28]:** We use a Tree-LSTM in place of our bidirectional LSTM encoder. Tree-LSTM transform LSTM from chain-like to tree-like structures. This bottom-up hierarchical tree-structured encoder composes the node state according to the input embedding and the node states of its child nodes in the expression tree.

From Table V we can see:

1) Models with Tree-LSTM perform worse than all other variants. We believe this is because Tree-LSTM transfers the node state in one direction. Instead of obtaining global information from the entire math expression, each node only obtains information from its own subtree.
2) Models with GCN show competitive performance compared to models without GCN, and show improvements in some metrics such as BLEU4 and CIDEr. We believe that both BiLSTM and GCN can capture the structure and global semantic information of the math expression. In this article, we use BiLSTM because it is simpler.
3) With adjacent node embeddings from math expression trees, MWPGen and MWPGen-base achieved competitive scores on automatic metrics, while also achieving higher accuracy on the problem solver. This shows

TABLE VI
ABLATION STUDY OF THE TOPIC-EXPRESSION CO-ATTENTION MECHANISM. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	BLEU1	BLEU2	BLEU3	BLEU4	Rouge_L	CIDEr	Expression	Answer
MWPGen-base (seq2seq)	0.5654	0.4440	0.3632	0.3032	0.5657	2.3100	50.52%	60.75%
MWPGen-base with CoAtt	**0.5675**	**0.4454**	0.3632	0.3024	0.5651	2.2996	54.23%	62.80%
MWPGen without CoAtt	0.5548	0.4343	0.3546	0.2956	0.5603	2.2834	55.89%	65.35%
MWPGen	0.5658	0.4448	**0.3638**	**0.3037**	**0.5659**	**2.3166**	**57.71%**	**66.43%**

TABLE VII
ABLATION STUDY OF REINFORCEMENT LEARNING. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

Model	BLEU1	BLEU2	BLEU3	BLEU4	Rouge_L	CIDEr	Expression	Answer
MWPGen-base (seq2seq)	0.5654	0.4440	0.3632	0.3032	0.5657	2.3100	50.52%	60.75%
MWPGen-base with RL	0.5548	0.4343	0.3546	0.2956	0.5603	2.2834	52.98%	62.46%
MWPGen without RL	**0.5694**	**0.4475**	**0.3655**	**0.3049**	**0.5669**	2.3094	55.29%	64.23%
MWPGen	0.5658	0.4448	0.3638	0.3037	0.5659	**2.3166**	**57.71%**	**66.43%**

that the adjacent node embeddings introduce additional structural information, which helps to generate problems more related to expressions.

4) We also noticed that the model "MWPGen" performs worse than "MWPGen-base with Adjacent" on automatic metrics such as BLEU, and we will explain this situation in the following ablation study on reinforcement learning.

These observations further verifies the effectiveness of the adjacent node embeddings of expressions.

Effect of topic-expression co-attention: Our model implements topic-expression co-attention in the model to generate math word problems related to both topic words and math expressions. To better understand the effectiveness of the topic-expression co-attention mechanism, we conduct an ablation study on the topic-expression co-attention mechanism.

As shown in Table VI, with the topic-expression co-attention mechanism, the performance of MWPGen and MWPGen-base is improved. This result proves the effectiveness of this co-attention mechanism. We believe that the topic-expression co-attention mechanism enables the model to extract the correlated information between topic words and expressions, and generate problems related to both inputs. This demonstrates the beneficial effect of using the topic-expression co-attention mechanism.

Effect of reinforcement learning: To measure the effect of reinforcement learning, we conduct an ablation study: our proposed model MWPGen and vanilla seq2seq model MWPGen-base are trained without the perceptual loss, shown in Table VII. We can see that, with reinforcement learning, the MWPGen's BLEU4 points would be reduced to 0.3037, and similar reductions have also appeared on other automatic metrics. However, the performance of model on problem solving evaluations are greatly improved, with an increment of more than 2% expression accuracy and answer accuracy. Tables V and VI show similar results. The model "MWPGen" performs worse than "MWPGen-base with Adjacent" and "MWPGen-base with CoAtt" on automatic metrics, but it achieves higher problem solving accuracy.

We believe that reinforcement learning promotes the model to generate more complete and more solvable problems, but during this process reinforcement learning reduces the scores of some automatic evaluation metrics, such as BLEU, because the fine-tuning process may change some n-gram phrases that appear in the ground truth. Human evaluation also proves our point. In the human evaluation, MWPGen showed improvement in both fluency and expression accuracy, and showed a slight improvement on the completeness. Since the model with reinforcement learning improves the performance on problem solving metrics and achieves a competitive performance on the automatic evaluation metrics, we still use reinforcement learning in our proposed model.

TABLE VIII
ABLATION STUDY OF THE NUMBER OF KEYWORDS. THE BEST SCORE OF EACH METRIC IS HIGHLIGHTED IN BOLD.

	BLEU1	BLEU2	BLEU3	BLEU4	Rouge_L	CIDEr	Expression	Answer
0	0.2770	0.1686	0.1232	0.0966	0.3293	0.6772	**76.90%**	**84.67%**
1	0.4114	0.2884	0.2224	0.1782	0.4316	1.2118	71.80%	78.00%
2	0.4889	0.3600	0.2839	0.2305	0.4986	1.6912	65.18%	72.37%
3	0.5311	0.4039	0.3239	0.2665	0.5340	2.0079	61.40%	69.28%
4	0.5501	0.4264	0.3457	0.2866	0.5524	2.1758	57.99%	66.49%
5	**0.5658**	**0.4448**	**0.3638**	**0.3037**	**0.5659**	**2.3166**	57.71%	66.43%

Analysis about topic words of different lengths: To have a deeper understanding about the effect of the number of topic words, we reduce the amount of topic words used as input in the model. Table VIII shows the results of conducted ablation experiments. First, models that do not provide topic words will result in the worst performance. Here, the model performance on the automatic evaluation metrics is very poor, and the generated problems have an even higher answer accuracy than the ground truth. Without topic words, the model can only generate a limited number of math word questions based on math expressions. Even if these problems can be solved to obtain the original expressions, we cannot use these repeated problems to help teachers and expand the data set. Then, as the number of given topic words increases, the performance of the model on the automatic evaluation metrics also improves, while the model performance on the expression and answer accuracy decreases. This situation may come from having more initial information, which makes the generated problems

TABLE IX
EXAMPLES OF GENERATED PROBLEMS WITH DIFFERENT NUMBER OF TOPIC WORDS AS INPUT.

Topic words: quantity, store, balloons, red, yellow **Expression:** (N0 * N1) + N2 **Ground Truth:** There are N0 red balloons in the store. There are N2 more yellow balloons than N1 times the number of red balloons. How many yellow balloons are there? 商店 有 红 气球 N0 个 ， 黄 气球 比 红 气球 的 N1 倍 多 N2 个 ， 黄 气球 有 多少 个 ？
Num=0: A number is N2 more than N1 times N0, this number = ? 比 N0 的 N1 倍 多 N2 的 数 = ？ **Num=1:** How many with N2 =N0 * N1. 多少 和 N2 = N0 * N1 · **Num=2:** A store bought N0 boxes of apples. The number of pears they bought is N2 more boxes than N1 times the number of apples. How many boxes of pears did they buy? 商店 运 来 苹果 N0 箱 ， 运 来 的 梨 比 苹果 的 N1 倍 还 多 N2 箱 ， 运 来 梨 多少 箱 ？ **Num=3:** The store has N0 packs of colorful balloons, and there are N1 colorful balloons in each pack. With N2 white balloons, how many balloons are there in the store? 商店 有 花 气球 N0 包 ， 每包 N1 个 . 还有 白 气球 N2 个 ， 商店 有 气球 多少 个 ？ **Num=4:** There are N0 red balloons in the store. The number of yellow balloons is N1 times of red balloons. There are N2 blue balloons. How many yellow balloons and blue balloons are there? 商店 里 有 红 气球 N0 个 ， 黄 气球 的 个数 是 红 气球 的 N1 倍 . 商店 里 有 蓝 气球 N2 个 。 黄 气球 和 蓝 气球 一共 多少 个 ？ **Num=5:** There are N0 yellow balloons in the store. The number of red balloons is N2 more than N1 times of yellow balloons. How many red balloons are there in the store? 商店 有 黄 气球 N0 个 ， 红 气球 的 个数 比 黄 气球 的 N1 倍 多 N2 个 · 商店 有 红 气球 多少 个 ？

TABLE X
TWO CASES OF PROBLEM GENERATED BY MWPGEN COMPARED WITH OTHER BASELINES.

Case 1: Topic words: quantity, rabbit, white, black **Expression:** N1 * N0 **Ground Truth:** N0 of the number of white rabbits is equal to the number of black rabbits. There are N1 white rabbits. How many black rabbits are there? 白兔 只 数 的 N0 等于 黑 兔 的 只 数 ， 白兔 有 N1 只 ， 黑 兔 有 多少 只 ？
Seq2Seq: A number divided by N0 is equal to N1, the number = ? 一 个 数 除以 N0 等于 N1 ， 这个 数 = ？ **NQG++:** There are N0 white rabbits and N1 black rabbits. How many black rabbits are there? 白兔 有 N0 只 ， 黑 兔 有 N1 只 ， 黑 兔 有 多少 只 ？ **Result:** N1 **MAGNET:** The number of white rabbits is N0 of the black rabbits, and the number of white rabbits is N1 of gray rabbits. How many white rabbits are there? 白兔 的 是 黑 兔 的 N0 ， 白兔 的 只 数 是 灰 兔 的 N1 . 白兔 有 多少 只 ？ **Result:** Unsolvable **MWPGen:** The number of black rabbits is N0 of the white rabbits. If there are N1 white rabbits, how many black rabbits are there? 黑 兔 的 只 数 是 白兔 的 N0 . 白兔 有 N1 只 ， 黑 兔 有 多少 只 ？
Case 2: Topic words: engineering,road, construction, speed, truck **Expression:** (N0 / N1) * N2 **Ground Truth:** During road construction of UNK School, N0 tons of sand were transported by truck, which accounted for N1 of the daily sand consumption. Workers accelerated the speed of road construction, so the daily sand consumption increased by N2. How many tons of sand should the truck transport every day? 学校 UNK 的 工地 上 在 修路 ， 卡车 已 运来了 N0 吨 沙子 ，正好 占 了 一天 用 沙 量 的 N1 ， 为了 加快 进 度 ， 工人 们 加快 了 修路 的 速度 ， 工地 每天 的 用 沙 量 也 增加 了 N2 ， 卡车 每天 应 多 运 多少 吨 沙子 ？
Seq2Seq: A cow eats N0 kg of grass every day, which accounts for N1 of the total feed. After N2 days, how much feed did the cow eat? 一 头 奶牛 ， 每天 吃 N0 的 草 ， 占 所 吃 食物 总数 的 N1 ， 吃 了 N2 天 ， 这 头 头 一共 吃 多少 ？ **NQG++:** The road construction team is going to build a N0 meter long road. The plan is completed within N1 days. In fact, N2 meters are built every day. How many days will it actually be built? 修路 队 要 修 一条 长 N0 米 的 路 ， 计划 N1 天 修 完 ， 实际 每天 修 N2 米 ， 实际 多少 天 修 完 ？ **Result:** N0 / N2 **MAGNET:** Road construction team A builds a road at a speed of N0 km/h, and the construction is completed in N1 hours. Team B will take N2 more hours to build this road.How long is this road? 甲 修路 队 修 一段路 ， 速度 是 N0 千米 / 小时 ， N1 小时 修 完 . 如果 是 乙 队 ， 要 再 修 N2 小时 ， 一共 修 了 多少 千米 ？ **Result:** N0 * N1 **MWPGen:** The road construction team built a N0 meter long road in N1 days. At this speed, how many meters of road can be built in N2 days? 一 个 修路 队 要 修 N0 千米 的 路 ， 前 N1 天 修 了 全程 的 N2 ， 照 这样 的 速度 ， 还 需要 多少 天 才能 修 完 ？

more relevant to the topic words and thus closer to the ground truth. At the same time, more initial information makes the generated problems more diverse which leads to a decrease in their expression and answer accuracy.

An example for this effect can be seen in Table IX. We can see that the more topic words used as input, the more detailed the problem generated by the model. Problems with richer input contain more information and are closer to the ground truth. Human inspection of the results found that models without topic words as input tend to generate simple and limited types of math word problems.

However, if the model needs to provide a large number of topic words as input, there will be an additional workload for teachers and in the data construction process. Moreover, when the number of topic words reaches a certain value, the improvement of the model effect by increasing the number of topic words is not as obvious. Therefore, we set the length of a given topic word list in this task to 5.

G. Case Study

Table X lists two example of problems generated by MWPGen compared with other baselines.

In Case 1, Seq2Seq does not realize that the topic is about the number of rabbits, and therefore it generates a problem unrelated to the topic. The problem generated by NQG++ asks the number of black rabbits already given in this problem. The problem generated by MAGNET is incomplete and incorrect.

In Case 2, when the given topic is about engineering, Seq2Seq generates a problem about cow eating grass. The problems generated by NQG and MAGNET cannot be solved to obtain the original expression “(N0/N1)*N2”. In this question, ground truth is asking about the speed of road construction after acceleration. The problem generated by MWPGen is asking about the length of a road built at a certain speed. However, the problem generated by MWPGen is related to the topic and can be solved to generate the given expression.

From these cases, we can see that the problems generated by Seq2Seq, NQG++ and MAGNET are quite complete and smooth. However, some of these problems cannot be solved or do not correspond to the given expressions. Some of these problems are unrelated to the topic words. Instead, the proposed MWPGen can generate topic-related problems and these problems can be solved to obtain given expressions. These further verify the effectiveness of the topic-expression co-attention mechanism and reinforcement learning.

IV. Related Work

A. Math Word Problem Generation

Recently, many studies on natural language generation have attracted a lot of attention, such as machine translation [4], [5], [29], dialogue generation [7], [8], [30], reading comprehension problem generation [6], [31], [32],image caption generation [10], [33], [34].With the development of deep neural networks, the problem generation task uses neural network structures and has achieved remarkable results.

Zhou et al. propose a sequence-to-sequence model with attention mechanism and copy mechanism to generate questions for the text from SQuAD dataset [23]. They enrich the model with answer position and lexical features. Wu et al. propose a "read-attend-comment" procedure for news comment generation and formalize the procedure with a reading network and a generation network [35]. Zhao et al. propose a novel document-level approach for question generation by using multi-step recursive attention mechanism on the document and answer representation to extend the relevant context [36].

Math word problem generation is the inverse task of the math word problem solving. It can be widely used in artificial intelligence testing, data set construction and education scenarios [1], [2]. Zhou and Huang propose a GRU-based seq2seq model with a maxout layer and entity-enforced loss for generating math word problems [11]. This model encode topic words and math expressions separately, and use both the topic and expression information to generate problems. Liyanage and Ranathunga use a BiLSTM model with attention mechanism to generate math word problems for three different languages, English, Sinhala and Tamil [37]. However, the above methods have not effectively considered the characteristics of math word problem generation tasks.

In this paper, we propose a topic-expression co-attention mechanism to extract the correlated information between topic words and expressions. We also leverage the adjacent node embeddings in the math expression tree to capture the structure and global semantic information of the math expression.

B. Math Word Problem Solving

Solving math word problems has long been a very popular task [38], [39] and various methods have been proposed in the past few years [40], [41].Recent approaches to solve math word problems usually use the Seq2Seq model [4] with attention mechanism [5] and copy mechanism [16], [33]. These models are trained to generate math expressions, which can be executed to generate answers to questions [42], [43].For instance, DNS [3] first proposed a method based Seq2Seq model, which directly maps the problem to the corresponding expression. Liu et al. incorporated a tree structured model with an auxiliary stack that generates the math expression tree from top to bottom [12]. Xie and Sun proposed a seq2tree model to generate each node in the expression tree based on its parent node and left sibling tree [13]. Recently, many works that treat math word problems as graphs have also shown better performance. Zhang et al. connected each number in the problem with nearby nouns to enrich the problem representations [44]. Wu et al. connected words that belongs to the same category in the external knowledge base to capture common sense information [43]. Li et al. constructed an input graph from both the math problem and its corresponding dependency tree to incorporate structural information [45].

For question generation, some methods [46], [47] treat question answering (QA) and question generation as complementary tasks and jointly train these two tasks. Yuan et al. feed the generated problem to a QA system and use the performance of the QA system as a metric of the quality of the problem [48]. Li et al. jointly trained models on visual question answering and visual question generation tasks to leverage the complementary relationship between questions and answers in images [49]. Deng et al. propose a novel joint learning model to solve the task of community question answering and answer summary generation [50].

Inspired by these methods, we use the pre-trained GTS model [13] to measure our proposed model. We feed the generated problem to the GTS and use the results to measure the completeness and solvability of the generated problem.

C. Reinforcement Learning

Recently, some studies have applied reinforcement learning (RL) to natural language generation tasks [51]–[53].A variety of reinforcement learning methods have been proposed to further improve natural language generation learning by leveraging reward functions. Rennie et al. presented an optimization method called self-critical sequence training (SCST), which normalizes the rewards obtained by sampled sentences and inference sentences [18]. Chen et al. proposed a RL-based graph-to-sequence model. This model uses BLEU and word movers distance (WMD) as reward functions [54]. Wan et al. proposed a code summarization model based on an abstract syntax tree structure in a reinforcement learning framework, and used BLEU scores as reward [55]. For the math word problem generation task, we solve the generated problems and compare the results with the given math expressions as rewards and fine-tune our model by reinforcement learning, thereby improving the quality and solvability of the generated problem.

V. Conclusion

This paper proposed a novel model MWPGen for the math word problem generation task. Adjacent node embeddings in the expression tree were used as additional embeddings to capture the structure and global semantic information of the math expression. A topic-expression co-attention mechanism was proposed to effectively consider the correlated information between topic words and expressions. In addition, we used the quality and solvability of the generated math word problems as rewards and fine-tuned our model by reinforcement learning. Experimental results confirmed that the proposed model MWPGen can generate more complete and solvable problems than other baselines, and these problems are more related to the given topic words and mathematical expressions.

Acknowledgment

The authors wish to thank the anonymous reviewers for their helpful comments. This work was funded by China National Key R&D Program (No. 2018YFB1005104).

References

[1] O. Polozov, E. O'Rourke, A. M. Smith, L. Zettlemoyer, S. Gulwani, and Z. Popović, "Personalized mathematical word problem generation," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[2] R. Koncel-Kedziorski, I. Konstas, L. Zettlemoyer, and H. Hajishirzi, "A theme-rewriting approach for generating algebra word problems," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1617–1628.

[3] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 845–854.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[6] X. Du, J. Shao, and C. Cardie, "Learning to ask: Neural question generation for reading comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1342–1352.

[7] B. Pan, H. Li, Z. Yao, D. Cai, and H. Sun, "Reinforced dynamic reasoning for conversational question generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2114–2124.

[8] J. Wang, J. Liu, W. Bi, X. Liu, K. He, R. Xu, and M. Yang, "Improving knowledge-aware dialogue generation via knowledge base question answering," *arXiv preprint arXiv:1912.07491*, 2019.

[9] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 1587–1596.

[10] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing, "Recurrent topic-transition gan for visual paragraph generation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3362–3371.

[11] Q. Zhou and D. Huang, "Towards generating math word problems from equations and topics," in *Proceedings of the 12th International Conference on Natural Language Generation*, 2019, pp. 494–503.

[12] Q. Liu, W. Guan, S. Li, and D. 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 (EMNLP-IJCNLP)*, 2019, pp. 2370–2379.

[13] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 5299–5305.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[16] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016. [Online]. Available: http://dx.doi.org/10.18653/v1/P16-1014

[17] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[18] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7008–7024.

[19] Z. Dong, Q. Dong, and C. Hao, "Hownet and the computation of meaning," 2006.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," pp. 1243–1252, 2017.

[23] Q. Zhou, N. Yang, F. Wei, C. Tan, H. Bao, and M. Zhou, "Neural question generation from text: A preliminary study," in *National CCF Conference on Natural Language Processing and Chinese Computing*. Springer, 2017, pp. 662–671.

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[25] C.-Y. Lin and F. Och, "Looking for a few good metrics: Rouge and its evaluation," in *Ntcir Workshop*, 2004.

[26] R. Vedantam, C. Lawrence Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4566–4575.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[28] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," pp. 1556–1566, 2015.

[29] J. Zhang, H. Luan, M. Sun, F. Zhai, J. Xu, M. Zhang, and Y. Liu, "Improving the transformer translation model with document-level context," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 533–542.

[30] S. Wu, Y. Li, D. Zhang, Y. Zhou, and Z. Wu, "Diverse and informative dialogue generation with context-specific commonsense knowledge awareness," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 5811–5820.

[31] Y. Zhao, X. Ni, Y. Ding, and Q. Ke, "Paragraph-level neural question generation with maxout pointer and gated self-attention networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3901–3910.

[32] B. Liu, M. Zhao, D. Niu, K. Lai, Y. He, H. Wei, and Y. Xu, "Learning to generate questions by learningwhat not to generate," in *The World Wide Web Conference*, 2019, pp. 1106–1118.

[33] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[34] Z.-J. Zha, D. Liu, H. Zhang, Y. Zhang, and F. Wu, "Context-aware visual policy network for fine-grained image captioning," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[35] Z. Yang, C. Xu *et al.*, "Read, attend and comment: A deep architecture for automatic news comment generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 5080–5092.

[36] L. A. Tuan, D. J. Shah, and R. Barzilay, "Capturing greater context for question generation," *arXiv preprint arXiv:1910.10274*, 2019.

[37] V. Liyanage and S. Ranathunga, "Multi-lingual mathematical word problem generation using long short term memory networks with enhanced input features," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 4709–4716.

[38] C. R. Fletcher, "Understanding and solving arithmetic word problems: A computer simulation," *Behavior Research Methods, Instruments, &amp; Computers*, vol. 17, no. 5, pp. 565–571, 1985.

[39] Y. Bakman, "Robust understanding of word problems with extraneous information," *arXiv preprint math/0701393*, 2007.

[40] D. Huang, S. Shi, C.-Y. Lin, and J. Yin, "Learning fine-grained expressions to solve math word problems," in *EMNLP*, 2017, pp. 805–814.
[41] S. Roy and D. Roth, "Mapping to declarative knowledge for word problem solving," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 159–172, 2018.
[42] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, "Program induction by rationale generation: Learning to solve and explain algebraic word problems," in *ACL*, vol. 1, 2017, pp. 158–167.
[43] Q. Wu, Q. Zhang, J. Fu, and X.-J. Huang, "A knowledge-aware sequence-to-tree network for math word problem solving," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 7137–7146.
[44] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim, "Graph-to-tree learning for solving math word problems," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 3928–3937.
[45] S. Li, L. Wu, S. Feng, F. Xu, F. Xu, and S. Zhong, "Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 2841–2852.
[46] T. Wang, X. Yuan, and A. Trischler, "A joint model for question answering and question generation," *arXiv preprint arXiv:1706.01450*, 2017.
[47] D. Tang, N. Duan, T. Qin, Z. Yan, and M. Zhou, "Question answering and question generation as dual tasks," *arXiv preprint arXiv:1706.02027*, 2017.
[48] X. Yuan, T. Wang, C. Gulcehre, A. Sordoni, P. Bachman, S. Zhang, S. Subramanian, and A. Trischler, "Machine comprehension by text-to-text neural question generation," in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017, pp. 15–25.
[49] Y. Li, N. Duan, B. Zhou, X. Chu, W. Ouyang, X. Wang, and M. Zhou, "Visual question generation as dual task of visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6116–6124.
[50] Y. Deng, W. Lam, Y. Xie, D. Chen, Y. Li, M. Yang, and Y. Shen, "Joint learning of answer selection and answer summary generation in community question answering." in *AAAI*, 2020, pp. 7651–7658.
[51] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 1192–1202.
[52] S. Narayan, S. B. Cohen, and M. Lapata, "Ranking sentences for extractive summarization with reinforcement learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 1747–1759.
[53] R. Csáky, P. Purgai, and G. Recski, "Improving neural conversational models with entropy-based data filtering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 5650–5669.
[54] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," 2019.
[55] Y. Wan, Z. Zhao, M. Yang, G. Xu, H. Ying, J. Wu, and P. S. Yu, "Improving automatic source code summarization via deep reinforcement learning," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 397–407.

Qinzhuo Wu received her master degree in software engineering from Peking University. She is a PhD student at Fudan of computer science at Fudan University, Shanghai, China. Her research interests include natural language processing and information retrieval.

Qi Zhang received the PhD degree in computer science from Fudan University. He is an associate professor of computer science at Fudan University, Shanghai, China. His research interests include natural language processing and information retrieval.

Xuanjing Huang received the PhD degree in computer science from Fudan University. She is a professor of computer science at Fudan University, Shanghai, China. Her research interests include natural language processing and information retrieval.