Towards Better Quantity Representations for Solving Math Word Problems

RUNXIN SUN, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
SHIZHU HE, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
JUN ZHAO, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
KANG LIU*, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China, School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China, and Shanghai Artificial Intelligence Laboratory, Shanghai, China

Solving a math word problem requires selecting quantities in it and performing appropriate arithmetic operations to obtain the answer. For deep learning-based methods, it is vital to obtain good quantity representations, i.e., to selectively and emphatically aggregate information in the context of quantities. However, existing works have not paid much attention to this aspect. Many works simply encode quantities as ordinary tokens, or use some implicit or rule-based methods to select information in their context. This leads to poor results when dealing with linguistic variations and confounding quantities. This paper proposes a novel method to identify question-related distinguishing features of quantities by contrasting their context with the question and the context of other quantities, thereby enhancing the representation of quantities. Our method not only considers the contrastive relationship between quantities, but also considers multiple relationships jointly. Besides, we propose two auxiliary tasks to further guide the representation learning of quantities: 1) predicting whether a quantity is used in the question; 2) predicting the relations (operators) between quantities given the question. Experimental results show that our method outperforms previous methods on SVAMP and ASDiv-A under similar settings, even some newly released

*Corresponding Author

Authors' Contact Information: Runxin Sun, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, Beijing, China; e-mail: vercosun@outlook.com; Shizhu He, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, Beijing, China; e-mail: shizhu.he@nlpr.ia.ac.cn; Jun Zhao, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, Beijing, China; e-mail: jzhao@nlpr.ia.ac.cn; Kang Liu, The Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, Beijing, China and School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, Beijing, China and Shanghai Artificial Intelligence Laboratory, Shanghai, Shanghai, China; e-mail: kliu@nlpr.ia.ac.cn.

ACM 2375-4702/2024/5-ART
https://doi.org/10.1145/3665644

strong baselines. Supplementary experiments further confirm that our method indeed improves the performance of quantity selection by improving the representation of both quantities and questions.

CCS Concepts: • **Computing methodologies** → *Natural language processing*.

Additional Key Words and Phrases: Math word problem solving, quantity representation, contrast, attention mechanism, auxiliary task

1 INTRODUCTION

A Math Word Problem (MWP) is a short narrative that describes a scenario involving several quantities, followed by a question that requires a series of mathematical operations to obtain the answer [1]. A typical example is shown in Figure 1. Due to its complexity, most existing works [37, 42, 44] predict the mathematical expression corresponding to the question instead of its answer directly. In recent years, solving MWP has received considerable attention, as it can serve as a good testbed for natural language understanding and multi-step reasoning against quantities [10, 11].

For deep learning-based methods, it is vital to obtain good quantity representations, which means associating each quantity with its related descriptive words according to their importance. For the example in Figure 1, the descriptive words related to “15" are “Julia", “played", “tag", “with", “kids", “on" and “Monday". Unfortunately, existing works have not paid much attention to this aspect. Many works, especially those using a Pre-trained Language Model (PLM, e.g., BERT [3]) as the encoder, replace each quantity with a dummy token (e.g., [NUM]) and then encode it like other ordinary tokens. We argue that it is far from enough. Li et al. [15] used multi-head attention [34] to model the relationship between *quantities and their context*, *quantity pairs*, and *quantities and questions*. However, this method is implicit because it still does not distinguish between quantity tokens (e.g., “15” in Figure 1) and other ordinary tokens in the sentence (e.g., “kids”). Moreover, it cannot consider these three relationships jointly, which we believe is crucial for assessing the importance of descriptive words. Zhang et al. [44] mined the descriptive words around quantities simply by a window rule and some dependency rules, then represented their relationship as a graph structure, which is a relatively “hard" way. In addition, Patel et al. [27] showed the vulnerability of existing models to linguistic variations and confounding terms (e.g., “18”), which also revealed their inability to model quantity representations effectively.

What factors influence the importance of descriptive words related to a quantity? We have the following observations. **First, their importance is related to the question.** Answering the question correctly is the goal of this task, and the key is to select the corresponding quantities in the narrative according to the question. For the example in Figure 1, to make “15" more easily to be selected by the question, it should be linked to the words “julia”, “played”, “with”, “kids”, “on” and “Monday”, which also appear in or are related to the words in the question. **Second, their importance is related to other quantities in the same problem.** By contrasting these descriptive words with those of other quantities, we can identify the unique features of the quantity and thus distinguish it from other quantities. Continuing with the previous example, we can see that “Monday” is a unique feature of “15” by contrast. Unfortunately, this contrasting relationship between quantities has been overlooked in previous studies.

To obtain better quantity representations, we propose a novel method called *Question-Quantity Contrastive Attention* (QQCA), which jointly considers the above two factors to assess the importance of descriptive words for quantities. Since quantity tokens do not contain any guiding information, the core idea of our method is to find words in the context of quantities by contrast that are similar to the words in the question, but dissimilar to the words in the context of other quantities. These words are considered to be the distinguishing features of quantities. Inspired by Liu et al. [20], our method consists of two steps. First, in the preprocessing stage, we mine the aforementioned words from the context of quantities through some rules and a PLM. Second, in the training/inference stage, we design an attention-based module to model the contrast process and use the results

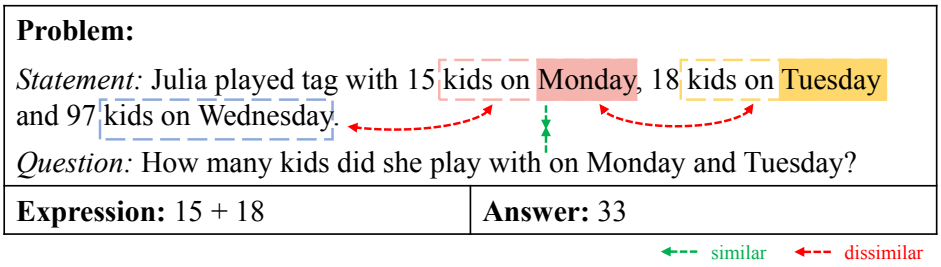

Fig. 1. A math word problem from SVAMP. The dashed boxes of different colors are the candidate windows corresponding to each quantity, and the highlighted parts are the unique features within the windows.

of the first step as prior knowledge to guide its training process. Moreover, we apply a similar method to identify selective features in the question to further improve the selection of quantities. In addition, to further guide the representation learning of quantities, **we propose two auxiliary tasks: 1) *Quantity Usage Prediction* predicting whether a quantity is used in the question; 2) *Quantity-pair Relation Prediction* predicting the relations (operators) between quantities given the question.** We conduct experiments on three widely used datasets: SVAMP [27], ASDiv-A [23], and MAWPS [10]. Experimental results show that our method achieves performance superior to all baselines (on SVAMP and ASDiv-A) or approximately equal to the state-of-the-art baseline (on MAWPS) under similar settings, even some newly released strong baselines [29, 36]. Furthermore, we validate the superiority of our method in selecting quantities according to the question by some additional experiments, which indicates to some extent that our method does indeed increase the variability of quantity representations and the selectivity of question representations.

Our contributions can be summarized in the following three points:

- We propose a novel method called Question-Quantity Contrastive Attention to identify distinguishing features in the context of quantities by contrast. To the best of our knowledge, our method is the first to consider the contrastive relationship between quantities, and the first to consider multiple relationships jointly.
- To further guide the representation learning of quantities, we propose two auxiliary tasks: 1) predicting whether a quantity is used in the question; 2) predicting the relations (operators) between quantities given the question.
- Our method outperforms previous methods on the SVAMP and ASDIV-A datasets under similar settings, even some newly released strong baselines.

2 RELATED WORK

2.1 Math Word Problem Solving

The research on solving math word problems has a long history, which can be traced back to the 1960s [1]. Early works mostly used hand-crafted rules or templates [1, 2]. Later works can be broadly divided into two categories: 1) statistical machine learning-based methods, which match problems with equation templates [6, 12, 24]; 2) semantic parsing-based methods, which parse natural language text into corresponding structured representations [28, 32, 47].

In recent years, deep learning-based methods have achieved considerable performance improvements and become dominant in this task. Wang et al. [37] made the first attempt by applying a vanilla sequence-to-sequence (seq2seq) model. Xie and Sun [42] proposed a tree-structured decoder to better utilize the structural properties of

expressions. Furthermore, Zhang et al. [44] proposed a graph encoder to inject knowledge through the graph structure. Since then, a series of works have been proposed to improve the encoder [19, 41] or introduce more knowledge [30, 39, 40]. Subsequently, due to the powerful capabilities of pre-trained language models, many works applied them to the encoding process and achieved impressive performance [33, 43]. Moreover, Liang et al. [16] proposed a series of task-specific self-supervised tasks and fine-tuned BERT [3] on this task for further usage. On the other hand, this has also led researchers to shift their focus toward improving the decoding process [7, 29, 35]. It is worth noting that some new research trends have emerged recently, such as using contrastive learning [18, 31, 45] and considering the diversity of arithmetic expressions [17, 25].

Our work builds on the idea of contrast, focusing on improving the encoding process to enhance the representation of quantities. A closely related work is Li et al. [15], which used multi-head attention to implicitly model the relationship between quantities and their context, quantity pairs, and quantities and questions. However, our work is quite different from theirs. First, our model focuses on quantities rather than treating them equally with ordinary tokens. Second, our model jointly considers the relationship between quantities and questions and the relationship between quantities and other quantities. Third, we designed two auxiliary tasks to guide representation learning. We hope our work could shed some light on obtaining better quantity representations.

3 PRELIMINARIES

3.1 Problem Definition

Formally, a Math Word Problem P is a sequence of n tokens $(w_1, w_2, ..., w_n)$, where each token w_i can be either a word from a natural language or a numerical value representing a quantity. Furthermore, it can be broken down into a statement $S = (w_1, w_2, ..., w_k)$ and a question $Q = (w_{k+1}, w_{k+2}, ..., w_n)$. Our goal is to generate a valid arithmetic expression E_p, which means that it logically fits the problem and its calculation result equals the answer. The vocabulary of E_p consists of three parts: the set of mathematical operators $V_{op} = \{+, -, *, /\}$, the set of numbers from the problem V_{num}, and the set of constants V_{con} (e.g., π). It should be noted that V_{num} varies with the problem.

3.2 Backbone Solver

We select Graph2Tree [44] enhanced with RoBERTa-base [21] as the backbone solver, which is a strong baseline widely adopted in previous works. It should be noted that our method is model-agnostic and thus can be transferred to any model. We leave it to future work.

Input Normalization and Encoder. First, as in previous works, we replace each number in the problem with a placeholder `[NUM]` to enhance the model's generalization ability. Patel et al. [27] pointed out that existing models can achieve reasonably high accuracy on MWPs when the question part is removed, indicating that they rely heavily on shallow heuristics from the statement without even looking at the question. To emphasize the question and thus alleviate this phenomenon, we break down each problem into two parts: a statement and a question, according to the previous definition. Then we add some special tokens and concatenate these two parts together as RoBERTa requires. The format of the input sequence is as follows:

$$X = \texttt{<s>}, S, \texttt{</s>}, \texttt{</s>}, Q, \texttt{</s>}$$

X is encoded with RoBERTa, followed by a bidirectional LSTM (bi-LSTM). Then we feed the question part to another bi-LSTM to get its sentence representation $\boldsymbol{h}_Q$, which we use to initialize the decoder. Subsequently, following Zhang et al. [44], we use Graph Neural Networks to encode two graphs: a *Quantity Cell Graph* connecting descriptive words with quantities, and a *Quantity Comparison Graph* representing the relative size relationship between quantities in numerical values. We take the encoding results as the final hidden representations and denote them as $\boldsymbol{H}_X$. Please refer to Zhang et al. [44] for more details.

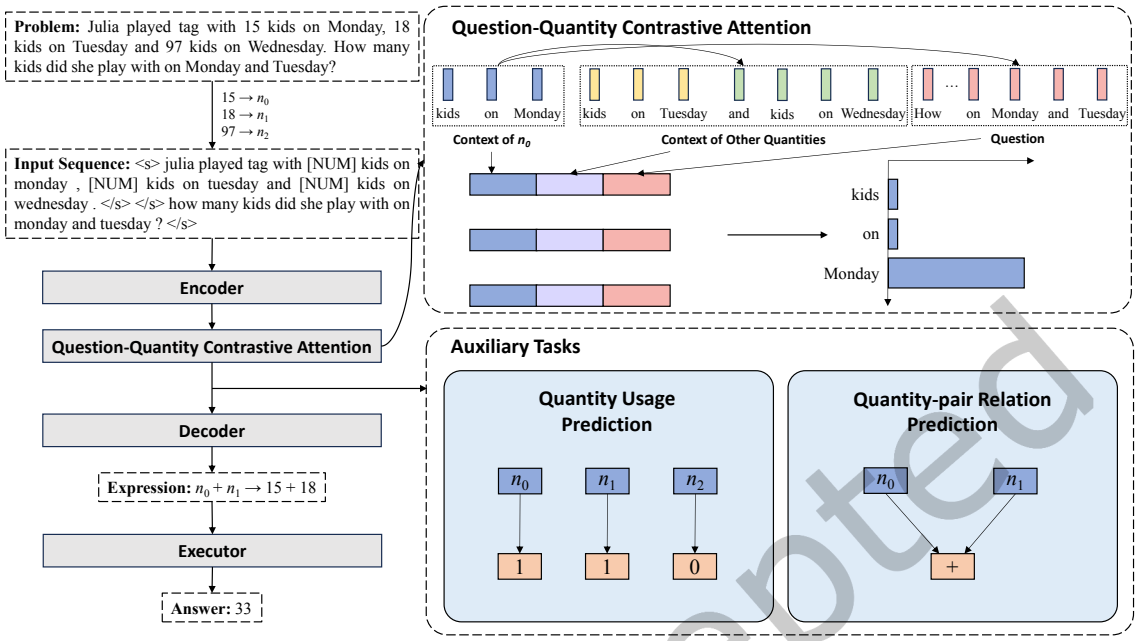

Fig. 2. An illustration of our method. It consists of two parts: (i) an attention-based module *Question-Quantity Contrastive Attention*, which identifies question-related distinguishing features of quantities by contrasting their context with the question and the context of other quantities; (ii) two auxiliary tasks: 1) *Quantity Usage Prediction* predicting whether a quantity is used in the question; 2) *Quantity-pair Relation Prediction* predicting the relations (operators) between quantities given the question.

Decoder. We use a tree-based decoder [42] to better model the structural properties of expressions, which is also widely adopted by previous works. Specifically, we first convert each expression into a binary tree where the root and intermediate nodes are arithmetic operators, and the leaf nodes are quantities. Then we generate it in the order of pre-order traversal. According to the previous definition, at each timestep t, the decoder will take one of the following three actions: generating an operator from V_{op}, generating a constant from V_{con}, or copying a quantity from V_{num}.

4 METHOD

4.1 Overview

As shown in Figure 2, our method consists of two parts. First, we propose a module called *Question-Quantity Contrastive Attention*. It identifies question-related distinguishing features of quantities by contrasting the words in their context with the words in the question and the words in the context of other quantities. Moreover, we apply a similar method to identify selective features in the question to further improve the selection of quantities. Second, we design two auxiliary tasks to further guide the representation learning of quantities. The first is *Quantity Usage Prediction*, which predicts whether a quantity should be used in the question. And the second is *Quantity-pair Relation Prediction*, which predicts the relations (operators) between quantities given the question.

4.2 Question-Quantity Contrastive Attention

In this step, we mine candidate windows around quantities that may contain descriptive words. Then we contrast the words in them with the words in the question and the words in other candidate windows to identify those words that help associate quantities with the question and distinguish them from other quantities. Our method consists of two steps, which are performed in the preprocessing stage and the training/inference stage, respectively. Next, we will introduce them in detail.

First, in the preprocessing stage, we assess the importance of candidate words through some rules and a PLM. We start by determining candidate windows for all quantities based on punctuation and the corresponding constituent syntax tree of the problem. Specifically, we split the problem into segments according to punctuation. Then, based on the constituent syntax tree, we split multiple quantities located in the same segment into disjoint segments, and select appropriate types of constituents to further shorten the length of segments containing quantities. All these disjoint segments containing a single quantity are candidate windows corresponding to their respective quantities. It is worth noting that multiple quantities are often presented in a parallel structure, that is, each quantity is located in a segment with a similar structure, and multiple such segments are arranged sequentially. Moreover, these segments depend on some shared constituents located before the first segment. The example in Figure 1 is a typical one, where "15", "18", and "97" are located in a parallel structure. It is easy to see that if we determine candidate windows based only on punctuation and certain types of constituents, those shared constituents will be assigned to the first candidate window, causing them to be highlighted as unique features of the first quantity after subsequent contrasts. In this case, we can identify the first parallel segment based on the pattern of the subsequent parallel segments, thus excluding the shared constituents from the first candidate window.

After determining the candidate windows, we assess the importance of the words in these windows by contrasting them with the words in the question and the words in other candidate windows. Specifically, we use a PLM RoBERTa-base [21] as the encoder to leverage its powerful understanding and representation capabilities, and use cosine distance to measure the semantic similarity between words. To strengthen the association between quantities and the question and distinguish them from other quantities, words with a higher similarity to the words in the question and a lower similarity to the words in other candidate windows should have higher weights. Therefore, we design the following formulas to measure the importance of words:

$$Sim(\boldsymbol{w}_i, \boldsymbol{w}_j) = \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|} \tag{1}$$

$$s_W^{i,Q} = \max_{w_j \in Q} Sim(\boldsymbol{w}_i, \boldsymbol{w}_j) \tag{2}$$

$$s_W^{i,O} = \max_{w_j \in O} Sim(\boldsymbol{w}_i, \boldsymbol{w}_j) \tag{3}$$

$$s_W^{i} = s_W^{i,Q}(1 - s_W^{i,O}), \tag{4}$$

where O are the words in other candidate windows. Formally, if $w_i \in W_k$, then $O = \bigcup_{j \neq k} W_j$. Furthermore, we normalize the scores of the words in each candidate window separately and consider them as a prior distribution:

$$p_{prior}^{k,i} = \frac{s_{W_k}^{i}}{\sum_{w_j \in W_k} s_{W_k}^{j}}. \tag{5}$$

In addition, we noticed that this task includes a crucial step of selecting quantities according to the question, which is essentially a process of matching the question with each quantity. If we want to improve this step, it would be wiser to improve both the quantity and the question representations so that the effect can be maximized. Therefore, we apply a similar method to identify selective features in a question by contrasting its words with

those in the candidate windows. We define selective features as clue words that only support selecting a particular quantity, and they often correspond to the unique features of that quantity. Analogous to the previous method, these words should have a higher similarity to the words in a particular candidate window and a lower similarity to the words in other candidate windows. From this, we get the following formulas:

$$s_Q^{i,W_j} = \max_{w_k \in W_j} Sim(\boldsymbol{w}_i, \boldsymbol{w}_k) \tag{6}$$

$$s_Q^{i,O_j} = \max_{l \neq j} \max_{w_k \in W_l} Sim(\boldsymbol{w}_i, \boldsymbol{w}_k) \tag{7}$$

$$s_Q^i = \max_j s_Q^{i,W_j}(1 - s_Q^{i,O_j}) \tag{8}$$

$$p_{prior}^{Q,i} = \frac{s_Q^i}{\sum_{w_j \in Q} s_Q^j}. \tag{9}$$

Second, in the training/inference stage, we propose an attention-based module to model the mechanism of the above contrast process. Again, we start with the quantities. We denote the hidden representations of the words in the i-th candidate window as $\boldsymbol{H}_{W_i}$, so the representations of the words in the other candidate windows are $\boldsymbol{H}_{O_i} = \bigcup_{j \neq i} \boldsymbol{H}_{W_j}$. Then we contrast $\boldsymbol{H}_{W_i}$ with $\boldsymbol{H}_Q$ and $\boldsymbol{H}_{O_i}$ separately through the attention mechanism:

$$\boldsymbol{H}_{W_i \to Q} = \text{Attention}(\boldsymbol{H}_{W_i}, \boldsymbol{H}_Q, \boldsymbol{H}_Q) \tag{10}$$

$$\boldsymbol{H}_{W_i \to O_i} = \text{Attention}(\boldsymbol{H}_{W_i}, \boldsymbol{H}_{O_i}, \boldsymbol{H}_{O_i}), \tag{11}$$

where $\boldsymbol{H}_Q$ are the representations of the words in the question. Subsequently, we concatenate the obtained results with the original representations of these words $\boldsymbol{H}_{W_i}$ and use a Multi-Layer Perceptron (MLP) to predict their importance scores:

$$s_{W_i}^j = \text{MLP}([\boldsymbol{h}_{W_i}^j; \boldsymbol{h}_{W_i \to Q}^j; \boldsymbol{h}_{W_i \to O_i}^j]). \tag{12}$$

Finally, we normalize these importance scores and use them as weights to aggregate the representations of the words in this candidate window:

$$A_{W_i} = \text{softmax}(S_{W_i}) \tag{13}$$

$$\boldsymbol{h}_i^D = \sum_{w_j \in W_i} \alpha_{W_i}^j \boldsymbol{h}_{W_i}^j. \tag{14}$$

The obtained result $\boldsymbol{h}_i^D$ is the question-related distinguishing feature of the i-th quantity. It is easy to see that we have essentially performed a "soft" selection of words in the candidate window using importance scores. To further guide the module to model the aforementioned bias, we use the normalized scores P_{prior}^i obtained in the preprocessing stage as the prior distribution and the KL divergence as its training objective:

$$\mathcal{L}_{quan} = \text{KL}(A_{W_i} || P_{prior}^i) = \sum_{w_j \in W_i} \alpha_{W_i}^j \log\left(\frac{\alpha_{W_i}^j}{p_{prior}^{i,j}}\right). \tag{15}$$

For the question, we apply a similar method to model its contrast process. We first contrast the words in the question $\boldsymbol{H}_Q$ with the words in all candidate windows $\boldsymbol{H}_W = \bigcup_i \boldsymbol{H}_{W_i}$ through the attention mechanism:

$$\boldsymbol{H}_{Q \to W} = \text{Attention}(\boldsymbol{H}_Q, \boldsymbol{H}_W, \boldsymbol{H}_W). \tag{16}$$

Then we predict the importance scores of the words in the question and use them as weights to aggregate these words into a single selective feature:

$$s_Q^i = \text{MLP}([\boldsymbol{h}_Q^i; \boldsymbol{h}_{Q\to W}^i]) \tag{17}$$

$$A_Q = \text{softmax}(S_Q) \tag{18}$$

$$\boldsymbol{h}_Q^S = \sum_{w_i \in Q} \alpha_Q^i \boldsymbol{h}_Q^i. \tag{19}$$

Finally, we define its training objective as the KL divergence from the prior distribution:

$$\mathcal{L}_{ques} = \text{KL}(A_Q || P_{prior}^Q) = \sum_{w_i \in Q} \alpha_Q^i \log\left(\frac{\alpha_Q^i}{p_{prior}^{Q,i}}\right). \tag{20}$$

As for fusing the above results with the original features, we follow Lu et al. [22] and adopt a gating mechanism to achieve it adaptively. We apply the same method to quantities and questions. Here we take the i-th quantity as an example. Specifically, we first project its original feature $\boldsymbol{h}_i^O$ and its distinguishing feature $\boldsymbol{h}_i^D$ into a universal space:

$$\boldsymbol{h'}_i^O = f_{Orig\to U}(\boldsymbol{h}_i^O) \tag{21}$$

$$\boldsymbol{h'}_i^D = f_{Dist\to U}(\boldsymbol{h}_i^D), \tag{22}$$

where $f_{Orig\to U}(\cdot)$ and $f_{Dist\to U}(\cdot)$ are linear layers with a nonlinear activation function. Then we predict their corresponding gate scores:

$$\tilde{\boldsymbol{g}}_i^j = f_{U\to G}(\boldsymbol{h'}_i^j), j \in \{O, D\} \tag{23}$$

$$\boldsymbol{g}_i^j = \frac{\exp(\tilde{\boldsymbol{g}}_i^j)}{\sum_k \exp(\tilde{\boldsymbol{g}}_i^k)}, \tag{24}$$

where $f_{U\to G}(\cdot)$ is a linear layer with a nonlinear activation function, and $\tilde{\boldsymbol{g}}_i^j$ and $\boldsymbol{g}_i^j$ have the same dimensions as $\boldsymbol{h'}_i^j$. Finally, we combine all the above results to obtain the final representation:

$$\boldsymbol{h}_i = \boldsymbol{g}_i^O \odot \boldsymbol{h'}_i^O + \boldsymbol{g}_i^D \odot \boldsymbol{h'}_i^D, \tag{25}$$

where $\odot$ is element-wise multiplication.

4.3 The Design of Auxiliary Tasks

To further guide the representation learning of quantities, we design two auxiliary tasks by mining and decomposing the generation target.

Quantity Usage Prediction. It predicts whether a quantity should be used in the question. In fact, quantities with similar meanings may have different usage statuses. For the example in Figure 1, "18" and "97" are two quantities with similar meanings (they both represent the number of kids with whom Julia played tag), but the former is used in the question while the latter is not. It is easy to see that this task essentially helps guide the model to capture the differences between quantities ("18" and "97" correspond to "Tuesday" and "Wednesday", respectively) and their associations with the question ("Tuesday" is also mentioned in the question).

Next, we present the implementation details. For each quantity, we concatenate its hidden representation $\boldsymbol{h}_i$ with the question representation $\boldsymbol{h}_Q$ as input, then use an MLP to predict its probability of being used:

$$p_{use}^i = \sigma(\text{MLP}([\boldsymbol{h}_i; \boldsymbol{h}_Q])), \tag{26}$$

where σ is the sigmoid function. The training objective is the binary cross-entropy loss, which has the following form:

$$\mathcal{L}_{use} = -\frac{1}{M}\sum_{i=1}^{M} y_{use}^{i}\log(p_{use}^{i}) + (1 - y_{use}^{i})\log(1 - p_{use}^{i}), \tag{27}$$

where y_{use}^{i} is the label indicating whether the i-th quantity is used (1 means it is used and 0 otherwise), and M is the total number of quantities in the dataset.

Quantity-pair Relation Prediction. It predicts the relations (operators) between quantities given the question, which is essentially a one-step decomposition of the generation target. By directly using the encoded representations to predict this task, we can guide the model to capture the associations between quantities in the encoding stage, such as associating an item's unit value and quantity to obtain its total value (via multiplication). Moreover, this task ignores the decoding order, which helps the model to improve its compositional generalization ability, that is, generating more complex structures by composing substructures [13]. It is worth noting that, for the sake of simplification, we only consider the case where two quantities are directly associated by an operator and ignore cases involving parentheses. This is because the association between a quantity and a parenthesis may not be decomposed into associations between the quantity and each quantity in the parenthesis. Consider the following example:

Problem: After resting they decided to go for a swim. If the depth of the water is 10 times Dean's height and he stands at 9 feet, how much deeper is the water than Dean's height?
Expression: (10 * 9) - 9

Fig. 3. A MWP example whose corresponding expression contains parentheses. We cannot remove the parentheses in this example because the "9" after the minus sign is associated with the entire expression inside the parentheses before the minus sign, but it has no direct association with each individual quantity inside the parentheses.

Although there is indeed an association (subtraction) between "10 * 9" and "9", it is hard to say that there are meaningful associations between "10" and "9" and between "9" and "9", respectively.

As for the implementation details, similarly, we concatenate the representations of each pair of quantities $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ with the question representation $\boldsymbol{h}_Q$ as input, then use an MLP to predict the probabilities of relations:

$$P_R^{i,j} = \text{softmax}(\text{MLP}([\boldsymbol{h}_i; \boldsymbol{h}_j; \boldsymbol{h}_Q])), \tag{28}$$

where R is the set of relations. In particular, for each pair of quantities i and j, we predict the relations between (i, j) and (j, i) considering the effect of the input (operand) order. Specifically, these two relations are exactly the same for operators satisfying the commutative law, and different otherwise. The training objective is defined as follows, which is a standard cross-entropy loss:

$$\mathcal{L}_{rel} = -\frac{1}{2N}\sum_{(i,j)}\sum_{r\in R} y_r^{i,j}\log(p_r^{i,j}) + y_r^{j,i}\log(p_r^{j,i}), \tag{29}$$

where $y_r^{i,j}$ is the label indicating whether the relation between (i, j) is r (1 means yes and 0 otherwise), and N is the total number of quantity pairs having a direct relation.

Dataset	Size	Avg Prob Len	#Avg Quan in Prob	CLD	#Eq Templates	#Avg Ops
SVAMP	1000	34.69	2.82	0.22	26	1.24
ASDiv-A	1218	32.29	2.36	0.50	19	1.23
MAWPS	1987	30.32	2.48	0.26	39	1.78

Table 1. Dataset Statistics. "Avg Prob Len" means the average problem length. "#Avg Quan in Prob" means the average number of quantities in a problem. "CLD" means Corpus Lexicon Diversity [23]. "#Eq Templates" means the number of equation templates. "#Avg Ops" means the average number of operators in an expression.

4.4 Training Objective

To sum up, the training objective of the model is:

$$\mathcal{L} = \mathcal{L}_{gen} + \lambda_1 \mathcal{L}_{quan} + \lambda_2 \mathcal{L}_{ques} + \lambda_3 \mathcal{L}_{use} + \lambda_4 \mathcal{L}_{rel}, \tag{30}$$

where λ_1, λ_2, λ_3 and λ_4 are hyperparameters, and $\mathcal{L}_{gen}$ is the negative log-likelihood loss used to train autoregressive generation. The form of $\mathcal{L}_{gen}$ is as follows:

$$\mathcal{L}_{gen} = -\frac{1}{D} \sum_{i=1}^{D} \sum_{t=1}^{T} \log p(y_t | y_{<t}, P), \tag{31}$$

where D is the total number of examples of the dataset, and T is the length of the gold expression.

5 EXPERIMENTS

5.1 Experimental Setup

Datasets. We conduct experiments on three widely used datasets: SVAMP [27], ASDiv-A [23], and MAWPS [10]. Their detailed statistics are shown in Table 1. For SVAMP, we combine MAWPS and ASDiv-A as the training set following Patel et al. [27] and average the performance over four random seeds. For ASDiv-A and MAWPS, we evaluate the performance through 5-fold cross-validation following previous works. We use *Value Accuracy (Val Acc.)* as the evaluation metric, which checks whether the calculation result of the predicted expression is equal to the answer. Rather than directly comparing to the annotated expression, it has the advantage that expressions that are formally different but essentially the same as the annotated expression will not be judged as wrong. Therefore, it is widely adopted by previous works.

Baselines. We select several classic and strong models as our baselines. **GROUP-ATT** [15] uses multi-head attention to extract various types of features of MWPs. **GTS** [42] adopts a Goal-driven Tree-Structured decoder to better model the structural properties of expressions. **Graph2Tree** [44] combines the previous tree-based decoder with a graph-based encoder to inject quantity-related relations into the model. It is worth noting that, as in many existing works [7, 27], we use RoBERTa-base as the encoder to enhance the performance of the previous two models. **DeductReasoner** [7] generates the tree structure of expressions in a bottom-up order and implements the reuse of substructures. **BERT-TECL** [31] applies Contrastive Learning to facilitate representation learning and utilizes Textual information to Enhance the mining of hard positive/negative examples. Furthermore, **CANTOR** [29] and **MsAT-DeductReasoner** [36] are two newly released strong baselines, which are state-of-the-art models at the same scale on SVAMP and ASDiv-a (and MAWPS), respectively. In addition, we also select some strong baselines on the MAWPS dataset. For example, **ATHENA** [8] simulates the process of human thinking (reasoning) by enumerating and merging multiple reasoning paths (sub-expressions). **Multi-View** [45] enhances the consistency of the representation space by aligning the representations of an arithmetic expression from different views (traversal orders). **Exp-Tree** [46] employs a layer-wise parallel decoding strategy, i.e., generating

Model	SVAMP	ASDiv-A	MAWPS
Large language models	(text-davinci-002) 63.7[a]	(code-davinci-002) 86.7[b]	(WizardMath 70B) 86.2[c]
GROUP-ATT	21.5†	61.0†	76.1
RoBERTa-GTS	41.0	81.2	88.5
RoBERTa-Graph2Tree	43.8	82.2	88.7
DeductReasoner	45.0	-	92.0
BERT-TECL	-	74.6	-
CANTOR	49.6	-	-
MsAT-DeductReasoner	48.9	87.5	**94.3**
ATHENA	45.6	86.4	92.2
Multi-View	-	-	92.3
Exp-Tree	-	-	92.3
Base	47.5 (± 1.3)	85.0	90.8
Ours	**50.4** (± 1.1)	**87.8**	94.2

Table 2. Performance (Val Acc.) of different models on SVAMP, ASDiv-A and MAWPS. The results of LLMs are from: *a*: Kojima et al. [9], *b*: Diao et al. [4], *c*: Gou et al. [5]. †: We report the reproduced results from Lan et al. [14].

multiple independent expressions in one layer of the expression tree at once. It significantly reduces the decoding steps of the expression tree, thereby enhancing the capacity for generating complex expressions.

Implementation Details. We implement our model using PyTorch [26] and train it on a Linux platform with NVIDIA RTX3090 graphics cards. We initialize RoBERTa-base with weights from HuggingFace's Transformers [38]. We set the hidden size to 384 (half of RoBERTa-base) and the number of layers of all LSTMs to 1. Following Zhang et al. [44], we use a two-layer GCN to encode each graph separately. For our module (QQCA), all attentions are single-head dot-product attention. The number of layers of all MLPs is set to 2, and their intermediate activation function is Tanh. We set both λ_3 and λ_4 to 1 and perform grid search on λ_1 and λ_2. The value set for both is $\{0.1, 0.2, 0.3, 0.5, 0.7, 1\}$. We choose Adam with weight decay as the optimizer, where the decay factor is set to 1e-5. The learning rate is 1e-5 for the pre-trained word embedding and 1e-3 for the rest. We train our model for 70 epochs with a batch size of 8.

5.2 Main Results

Table 2 compares the performance of our model with the base model and all baselines on the three datasets. It should be noted that the base model we employed (Base in the table) is exactly the same as RoBERTa-Graph2Tree in model structure. We mainly made the following two modifications: 1) We fixed a bug in the implementation of Patel et al. [27] regarding the position of quantity tokens. 2) To make the model focus more on the target to be solved, we initialize the decoder with the hidden representation of the last sentence (i.e., the question) instead of the entire problem.

From the results, we can observe that our model not only significantly outperforms the base model (SVAMP +2.9, ASDiv-A +2.8, MAWPS +3.4), but also achieves performance that is superior to all baselines (SVAMP +0.8, ASDiv-A +0.3) or approximately equal to the state-of-the-art baseline (94.2 vs. 94.3 on MAWPS) on the three datasets. This demonstrates the effectiveness of our method. It is worth noting that CANTOR is committed to improving the decoding side, which uses a Directed Acyclic Graph (DAG) to consider multiple possible reasoning

Model	Val Acc.	Δ
Ours	**50.4** (± 1.1)	
-QQCA	49.9 (± 1.0)	-0.5
-$\mathcal{L}_{use}$	48.0 (± 0.6)	-2.4
-$\mathcal{L}_{rel}$	48.8 (± 0.9)	-1.6

Table 3. Ablation results on different components of our method. "-" denotes that we remove the component.

Model	Val Acc.	Δ
Ours - QQCA	49.9 (± 1.0)	
+Modules	47.4 (± 0.6)	-2.5
+Modules & $\mathcal{L}_{quan}$	48.0 (± 0.9)	+0.6
+Modules & $\mathcal{L}_{ques}$	48.6 (± 1.5)	+1.2
+Modules & $\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$ (QQCA)	**50.4** (± 1.1)	**+3.0**

Table 4. Ablation results on different components of our QQCA module. "+Modules" refers to adding only the modules introduced in Section 4.2 in the training/testing stage without any additional training objectives.

paths simultaneously and eliminate the restriction of a pre-defined decoding order. And MsAT-DeductReasoner not only employs a DAG-based decoder, but also appends a continual pre-training stage before downstream task fine-tuning to enhance the capabilities of a PLM related to solving MWPs. However, our method focuses on improving the encoding side for better quantity representations, which is orthogonal to theirs. In fact, our method can be combined with theirs to achieve further improvements. We leave it to future work.

In addition, we also compare the performance of our model with the zero-shot performance of some Large Language Models (LLM) [4, 5, 9]. We report the best results among them in Table 2. Surprisingly, although these LLMs have a clear advantage on SVAMP, they underperform our small model on ASDiv-A and MAWPS. We speculate that this is because although LLMs demonstrate powerful natural language understanding capabilities and robustness to language variations (on SVAMP), they do not have an advantage in problem solving (on ASDiv-A and MAWPS), i.e., generating arithmetic expressions under zero-shot setting. Considering that generating arithmetic expressions is essentially selecting quantities and predicting the relations (operators) between them, it is reasonable to believe that our method (more generally, the small model-based methods) can obtain better quantity representations than LLMs through some unique designs for the MWP solving task and fine-tuning the model on annotated task data. Moreover, due to the cost and efficiency advantages of small models, it is still worthwhile to conduct research on them.

5.3 Ablation Study

To validate the effectiveness of each part of our method, we conducted an ablation study on SVAMP, and the results are shown in Table 3. Here QQCA refers to the entire *Question-Quantity Contrastive Attention* module (including all modules and losses in Section 4.2). $\mathcal{L}_{use}$ and $\mathcal{L}_{rel}$ refer to the two auxiliary tasks *Quantity Usage Prediction* and *Quantity-pair Relation Prediction*, respectively.

From the results, we can observe that the model performance showed varying degrees of decline after removing each part separately, which proves their respective effectiveness. Besides, the improvement brought by QQCA is not as significant as the auxiliary tasks (0.5 compared to 2.4 and 1.6). We suspect that this may be due to the following two reasons. First, the fusion mechanism does not learn well enough. This may cause it to assign too

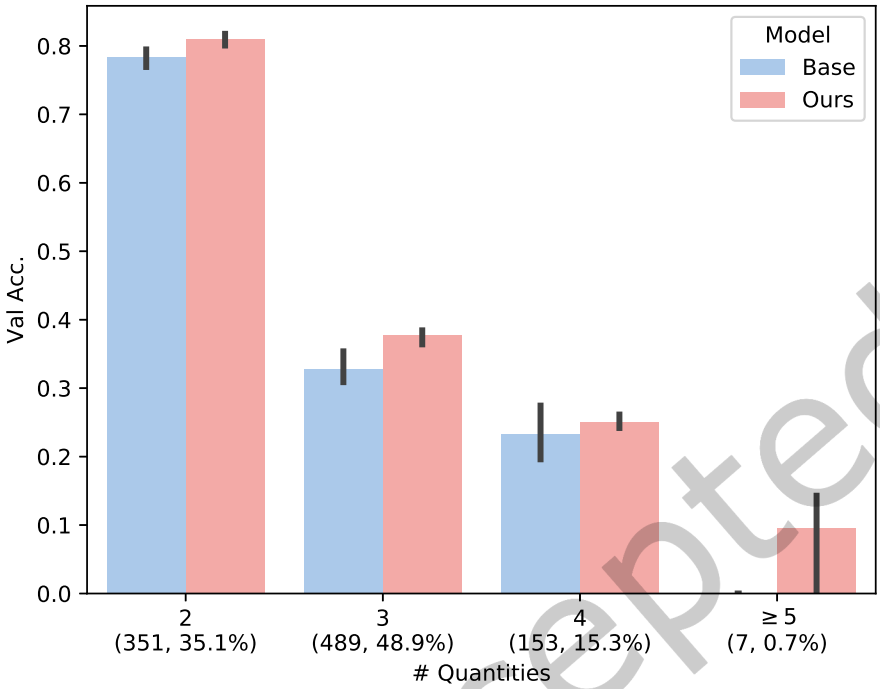

Fig. 4. Performance of the base model and our model with respect to the number of quantities in a problem. The content in parentheses is the number of problems belonging to that group and its proportion in the test set.

much weight to the newly obtained low-quality feature, thus reducing the quality of the final representation. Second, there is a difference in distribution between the training and test sets of SVAMP because its test set is obtained by manual adversarial modification from the sampled examples. This may cause the module to be undertrained because there are not enough examples in the training set that need to select or distinguish quantities.

Furthermore, we conducted another ablation study for a more in-depth analysis of our QQCA module. Table 4 presents the corresponding results. Here "+Modules" refers to the case where only the modules (parameters) are added without additional training objectives. $\mathcal{L}_{quan}$ and $\mathcal{L}_{ques}$ are the two training objectives based on prior distributions for quantities and questions, respectively. From the results, we can get the following observations. First, simply adding modules (that is, only increasing the number of parameters) did not bring any performance improvements but resulted in a significant decrease. We suspect there are two possible reasons: 1) Learning the fusion mechanism is quite tricky, which actually confirms our previous speculation to a certain extent; 2) The dataset size is too small, so increasing the parameters may exacerbate overfitting. Second, both of our training objectives lead to performance gains, which indicates their effectiveness and the high quality of prior distributions. Third, using both objectives simultaneously (+Modules & $\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$) is significantly more effective than using only the objective for quantities (+Modules & $\mathcal{L}_{quan}$, +3.0 vs. +0.6). This confirms our conjecture that improving both the quantity and the question representations can maximize the gains.

Model	Val Acc.	Δ
Base + Modules	22.5 (± 4.5)	
w/ the question	26.7 (± 2.2)	+4.2
w/ other quantities	26.6 (± 1.2)	+4.1
w/ the union ($\mathcal{L}_{quan}$)	**27.5** (± 2.8)	+**5.0**

Table 5. Ablation results on different relationships. "w/" denotes which relationships we consider when mining features in the context of quantities.

Model	Use Acc.	Δ
Base + Modules	27.2 (± 2.3)	
+$\mathcal{L}_{quan}$	30.1 (± 2.6)	+2.9
+$\mathcal{L}_{ques}$	29.0 (± 2.8)	+1.8
+$\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$	31.0 (± 1.7)	+3.8
+$\mathcal{L}_{use}$	31.5 (± 4.9)	+4.3
+$\mathcal{L}_{use}$ & $\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$	**35.3** (± 3.3)	+**8.1**

Table 6. Performance of different variants of our model in using quantities, which means selecting appropriate quantities to use for expression generation. *Use Acc.* check whether the quantities in the predicted expression are the same as those in the gold expression.

5.4 Supplementary Analysis

5.4.1 Analysis of the Number of Quantities. We group the problems in SVAMP by the number of quantities they contain and compare the performance of the base model and our model on each group. Figure 4 shows the corresponding results. From this, we can make the following observations. First, the performance of both models decreases as the number of quantities in the problem increases. It indicates that the capabilities of existing models are still too limited to solve more complex problems. Second, our model outperforms the base model for all groups. Furthermore, the increment of our model mainly comes from problems containing three quantities. Considering that all operators in the dataset are binary, it suggests that our model is better at selecting and distinguishing multiple quantities than directly predicting the relation between two quantities. This is also consistent with the original intent of our method.

5.4.2 Effects of Different Relationships and Their Union. In Section 4.2, we jointly consider two relationships when performing feature mining in the context of quantities: 1) the relationship between quantities and the question, and 2) the relationship between quantities and other quantities. We conducted experiments to investigate the effects of different relationships and their union. The results are shown in Table 5. It is worth noting that here we filtered SVAMP to remove problems that simply concatenate all quantities using the same operator to focus on evaluating the model's ability to select and distinguish quantities.

From the results, we can observe that 1) both relationships help to select and distinguish quantities (lines 2-3). It reveals that good quantity representations are not only related to the question but also to other quantities, where the latter have been overlooked in previous works. 2) The union of the two relationships increases the effect even more, proving the necessity of jointly considering both.

5.4.3 Evaluating the Usage of Quantities. In Section 4.2, we mentioned that this task includes a crucial step of selecting quantities according to the question, and we make improvements in representation from both quantities

Problem	Base	Ours
A waiter had n_0 customers. While n_1 customers left he got n_2 new customers. How many customers does he still have?	$n_0 - n_1 - n_2$	$n_0 - n_1 + n_2$
n_0 campers went rowing and n_1 campers went hiking in the morning. n_2 campers went rowing in the afternoon. How many campers went rowing in all?	$n_0 + n_1 + n_2$	$n_0 + n_2$
Rachel has n_0 apples trees. She picked n_1 apples from each of her trees. Now the trees have a total n_2 apples still on them. How many apples did Rachel pick in all?	$n_0 * n_1 + n_2$	$n_0 * n_1$

Table 7. Three examples from SVAMP demonstrating the superiority of our method. They respectively correspond to three characteristics of our method: 1) the distinguishability between quantities, 2) the selectivity of the question to quantities, 3) the indicativeness of quantities to the question.

and questions. Besides, in Section 4.3, we designed an auxiliary task that predicts whether a quantity is used in the question to further guide the representation learning of quantities and questions. To verify that these techniques have indeed improved this selection process, we compared the model's accuracy in quantity usage before and after using them. It means that all necessary quantities were used without using any noise terms. Specifically, we filter out examples in SVAMP where not all the quantities in the problem are used. We then check whether the quantities in the predicted expression for these examples are the same as those in the gold expression, ignoring the effect of operators and operation order (*Use Acc.*).

From the results, we can make the following observations. First, for our QQCA module (lines 2-4), our improvements to both quantity representations (+$\mathcal{L}_{quan}$) and question representations (+$\mathcal{L}_{ques}$) have worked, and the former is more effective. Furthermore, improving at both ends (+$\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$) yielded more significant improvements, confirming our previous speculation that the selection is a process of matching each quantity with the question. Second, the *Quantity Usage Prediction* auxiliary task (+$\mathcal{L}_{use}$) can also improve the process of selecting quantities according to the question, which is very straightforward. It also shows that relying solely on the end-to-end training paradigm is insufficient for the model to learn this step well. In addition, further improvements have been achieved by combining the above two technologies (+$\mathcal{L}_{use}$ & $\mathcal{L}_{quan}$ & $\mathcal{L}_{ques}$). We speculate that this is because our QQCA module considers the relationship between quantities and other quantities, while the auxiliary task does not.

5.5 Case Study

To demonstrate the superiority of our method more intuitively, we present three examples from SVAMP with the generated expressions of the base model and our model in Table 7. For the first example, all three quantities refer to the number of customers, but the states corresponding to these three groups of customers are different. However, the base model did not distinguish between n_1 and n_2 and assigned negative signs to both. In contrast, our model can find that the distinguishing feature of n_1 is "left" and that of N2 is "new" by contrasting their contexts, thus assigning different signs to them. For the second example, all three quantities refer to the number of campers, but the question asks for the total number of campers who went rowing. Our model successfully excluded n_1 from the generated expression, while the base model failed. We conjecture that it is because our method contrasts the context of quantities with the question, thus highlighting the crucial information "rowing" around n_0 and n_2. For the third example, the question is about the number of apples Rachel picked in all, where

the key is "pick" rather than "in all". Not surprisingly, the base model was tricked into generating an expression for the total number of apples. On the contrary, our model successfully identified "pick" as a key feature that helps to select quantities by contrasting the question with the context of quantities (inspired by "picked" around n_1), thus excluding the irrelevant term n_2 and generating the correct expression.

To sum up, these three examples intuitively and respectively demonstrate three characteristics of our method: 1) the distinguishability between quantities, 2) the selectivity of the question to quantities, and 3) the indicativeness of quantities to the question.

6 CONCLUSION

In this paper, we propose a novel method called Question-Quantity Contrastive Attention (QQCA). It identifies question-related distinguishing features of quantities by contrasting words in their context with words in the question and words in the context of other quantities. To the best of our knowledge, our method is the first to consider the contrastive relationship between quantities, and the first to consider multiple relationships jointly. Besides, we propose two auxiliary tasks to further guide the representation learning of quantities. Experimental results show that our method outperforms previous methods on SVAMP and ASDiv-A under similar settings. Ablation studies and supplementary analyses further illustrate the effectiveness of our method.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (No.2022ZD0160503) and the National Natural Science Foundation of China (No.62376270), Youth Innovation Promotion Association CAS.

REFERENCES

[1] Daniel Bobrow et al. 1964. Natural language input for a computer problem solving system. (1964).
[2] Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*. 303–316.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.
[4] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246* (2023).
[5] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452* (2023).
[6] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 523–533.
[7] Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: math word problem solving as complex relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 5944–5955.
[8] Jb. Kim, Hazel Kim, Joonghyuk Hahn, and Yo-Sub Han. 2023. ATHENA: Mathematical Reasoning with Thought Expansion. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 16315–16327. https://doi.org/10.18653/v1/2023.emnlp-main.1014
[9] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*. 22199–22213.
[10] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1152–1157.
[11] Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2022. Practice makes a solver perfect: Data augmentation for math word problem solvers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4194–4206.
[12] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 271–281.

[13] Yunshi Lan, Lei Wang, Jing Jiang, and Ee-Peng Lim. 2022. Improving compositional generalization in math word problem solving. *arXiv preprint arXiv:2209.01352* (2022).
[14] Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*. 13188–13190.
[15] Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 6162–6167.
[16] Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*. 997–1009.
[17] Zhenwen Liang, Jipeng Zhang, Lei Wang, Yan Wang, Jie Shao, and Xiangliang Zhang. 2022. Generalizing math word problem solvers via solution diversification. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*. 13183–13191.
[18] Zhenwen Liang, Jipeng Zhang, and Xiangliang Zhang. 2022. Analogical math word problems solving with enhanced problem-solution association. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 9454–9464.
[19] Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, and Shijin Wang. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *Proceedings of the 35th AAAI conference on artificial intelligence*. 4232–4240.
[20] Qian Liu, Dejian Yang, Jiahui Zhang, Jiaqi Guo, Bin Zhou, and Jian-Guang Lou. 2021. Awakening latent grounding from pretrained language models for semantic parsing. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 1174–1189.
[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[22] Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2019. Distilling discrimination and generalization knowledge for event detection via delta-representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 4366–4376.
[23] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 975–984.
[24] Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 2144–2153.
[25] Ansong Ni, Jeevana Priya Inala, Chenglong Wang, Alex Polozov, Christopher Meek, Dragomir Radev, and Jianfeng Gao. [n. d.]. Learning math reasoning from self-sampled correct and partially-correct solutions. In *The Eleventh International Conference on Learning Representations*.
[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2019).
[27] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems?. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2080–2094.
[28] Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics* 6 (2018), 159–172.
[29] Zhihong Shao, Fei Huang, and Minlie Huang. 2022. Chaining simultaneous thoughts for numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2533–2547.
[30] Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*. 2924–2934.
[31] Yibin Shen, Qianying Liu, Zhuoyuan Mao, Fei Cheng, and Sadao Kurohashi. 2022. Textual enhanced contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 4297–4307.
[32] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1132–1142.
[33] Minghuan Tan, Lei Wang, Lingxiao Jiang, and Jing Jiang. 2021. Investigating math word problems using pretrained multilingual language models. *arXiv preprint arXiv:2105.08928* (2021).
[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Proceedings of the 30th International Conference on Neural Information Processing Systems* (2017).
[35] Bin Wang, Jiangzhou Ju, Yang Fan, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2022. Structure-unified M-tree coding solver for math word problem. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 8122–8132.
[36] Tianduo Wang and Wei Lu. 2023. Learning multi-step reasoning by solving arithmetic tasks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 1229–1238.

[37] Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on Empirical Methods in Natural Language Processing*. 845–854.
[38] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: system demonstrations*. 38–45.
[39] Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuan-Jing Huang. 2020. 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*. 7137–7146.
[40] Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. 1473–1482.
[41] Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuan-Jing Huang. 2021. Math word problem solving with explicit numerical values. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 5859–5869.
[42] Zhipeng Xie and Shichao Sun. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems.. In *Ijcai*. 5299–5305.
[43] Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3384–3394.
[44] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3928–3937.
[45] Wenqi Zhang, Yongliang Shen, Yanna Ma, Xiaoxia Cheng, Zeqi Tan, Qingpeng Nong, and Weiming Lu. 2022. Multi-view reasoning: Consistent contrastive learning for math word problem. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. 1103–1116.
[46] Wenqi Zhang, Yongliang Shen, Qingpeng Nong, Zeqi Tan, Yanna Ma, and Weiming Lu. 2023. An Expression Tree Decoding Strategy for Mathematical Equation Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 439–456. https://doi.org/10.18653/v1/2023.emnlp-main.29
[47] Yanyan Zou and Wei Lu. 2019. Text2Math: End-to-end parsing text into math expressions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 5327–5337.

Received 7 August 2023; revised 4 February 2024; accepted 27 April 2024