IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 2 February 2023, accepted 13 March 2023, date of publication 15 March 2023, date of current version 22 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3257571

RESEARCH ARTICLE

Integrating Heterogeneous Graphs Using Graph Transformer Encoder for Solving Math Word Problems

SOYUN SHIN[1,2], JAEHUI PARK[1], AND MOONWOOK RYU[3]

[1]Department of Statistics, University of Seoul, Seoul 02504, South Korea
[2]AR-Bridge, Daejeon 34129, South Korea
[3]Content Research Division, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea

Corresponding authors: Jaehui Park (jaehui@uos.ac.kr) and Moonwook Ryu (moonwook@etri.re.kr)

This work was supported by the Culture, Sports and Tourism Research and Development Program through the Korea Creative Content Agency funded by the Ministry of Culture, Sports and Tourism, in 2022 (Project Name: Development of a Functional Content Platform for Stress Relief) (Contribution Rate: 100%) under Project R2020060003.

ABSTRACT This paper introduces a novel method that integrates structural information with training deep neural models to solve math word problems. Prior works adopt the graph structure to represent rich information residing in the input sentences. However, they lack the consideration of different relation types between other parts of the sentences. To provide various types of structural information in a uniform way, we propose a graph transformer encoder to integrate heterogeneous graphs of various input representations. We developed two types of graph structures. First, the *Dependency Graph* maintains long-distance lexical dependency between words and quantities. Second, the *Question Overlap Graph* captures the gist within the problem body. The two graphs are encoded as a single graph for graph transformation. Experimental results show that our method produces competitive results compared to the baselines. Our model outperforms state-of-the-art models in Equation and Answer accuracy near three percent in SVAMP benchmark. Moreover, we discuss that integrating different types of textual characteristics may improve the quality of mathematical logic inference from natural language sentences.

INDEX TERMS Math word problems, heterogeneous graphs, graph transformer networks, graph neural networks.

I. INTRODUCTION

Solving math word problems (MWP) requires a comprehensive understanding of the meaning of the natural language question and the mathematical knowledge in the given sentences. It is challenging because it is hard to derive mathematical knowledge from several given sentences. Utilizing the latent information from the sequences of words and numbers would be a key factor for developing accurate solvers. Due to the complexity of the problems, the class of MWP can be divided into arithmetic word problems, equation set problems, geometry word problems, and others. [1].

Table 1 shows an example of an arithmetic word problem. The problem comprises three parts: problem, equation, and answer. Pieces of evidence are described in the body, and a question sentence follows. Given the problem, the MWP solver should predict the equation, and then derives the answer. The equation is described as a sequence of operators and numbers. A final answer is a number. In the example, the number of birds in a tree is the variable we want to infer. The equation '∗ 1 12' is estimated based on the quantity words,

TABLE 1. An example of an arithmetic word problem.

Problem	
Body	Sally saw 1 dozen birds in a tree.
Question	How many birds did Sally see ?
Equation	
* 1 12	
Answer	
12	

The associate editor coordinating the review of this manuscript and approving it for publication was Arianna Dulizia.

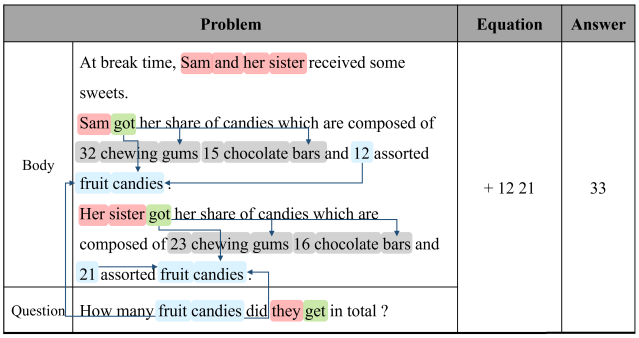


FIGURE 1. An example of MWP that contains various types of relationships.

'dozen.' Predicting a correct equation is challenging because the solver should understand the meaning of the mathematical operations and the words illustrating some quantities and their relationships. Figure 1 describes the associations between the words and the numbers that occurred at different parts of the sentences. For example, the quantity '12' indicates the number of 'fruit candies'. Also, the word 'they' refers to the two words 'Sam' and 'her sister,' which are occurred at different positions. Different types of associations are expressed in different colors. However, existing studies, for example, [2], may lack the consideration of structural information of problem body sentences. The sequence-to-sequence models may not be suitable for identifying the relationships between the quantity and the words to explain it. Moreover, the work [2] may ignore the connection between the word 'fruit candies' and the number '21' because they are not adjacent. To resolve this issue, the work, Graph-to-tree [3], adopts the graph structure to represent the input to generate an expression tree. Although the graph may represent rich structural information in the input sentences, it cannot distinguish the different relation types, such as dependency, adjacency, or ordering.

Automatic solving of MWP has been one of the most challenging tasks in artificial intelligence because the solver requires an ability to derive complex equations from natural language sentences. However, it could overfit to the limited set of training sentences. Therefore, traditional studies mainly focused on adapting existing rules and statistics to deduce the mathematical equation. However, with the increasing popularity of natural language processing (NLP) method based on machine learning (ML) studies, recent MWP studies have shown success in improving the ability to infer the sequence or the structure of mathematical symbols based on ML [2], [3], [4]. Recent studies utilize deep neural models [1], such as a sequence-to-sequence architecture to derive the equation [2] and Graph-to-tree approach [3] for representing the structural inputs. The recent improvements may be affected by deep learning, such as neural language models [5], [6], sequence learners [7], and attention mechanisms [8].

This paper proposes a novel graph transformer encoder to integrate heterogeneous graphs of various input representations. We presume that latent features can be easily captured by existing graph neural models, such as Graph-to-tree [3] with a graph transformer network [9]. The motivation of our method is to provide different types of structural information to a single training model in a uniform way. The encoder module accepts multiple edges with different types if they are represented in graph structures. The graph transformer encoder converts the various inputs into a single representation with deduced structure. We mean that the deduction identifies potential linkage between the nodes with no connections. We train the model with the new relationships, which may affect the quality of generated MWP solutions. For evaluation, we developed two types of graph structure, Dependency Graph, and Question Overlap Graph, and encoded them as a single input. The dependency Graph was constructed based on the directed binary grammatical relations between words. To extract the relations, we used SpaCy [10], a popular toolkit for NLP. The Question Overlap Graph was constructed based on the co-occurrence of words in the problem body and the question. A tree-based decoder generates the final equations by using these two graphs.

We summarize the main contributions as follows:

1) We propose a novel graph transformer encoder to integrate graphs for various input representations.

2) We evaluate the proposed encoder by proposing two types of graph structure, Dependency Graph and Question Overlap Graph.

3) An experimental study shows that our model yields state-of-the-art performance with equation accuracy and answer accuracy in benchmark datasets. Moreover, our model performs with robustness in the variational datasets.

An experimental study is described with several benchmark datasets. We evaluated the proposed method using three datasets: MAWPS, ASDiv-A, and SVAMP. We measure the answer and equation accuracy as two performance metrics. Experimental results show that our model yields state-of-the-art performance with the equation-accuracy of 88.54 and the answer-accuracy of 89.58. Moreover, we evaluated three quality measures, question sensitivity, reasoning ability, and structural invariance. As adding variations to the problems, our model shows superior performance, demonstrating the robustness of our model in complex problem settings.

The organization of this paper is as follows. Section II presents the related studies in the field of MWP. Section III introduces our proposed methods and training algorithms. Section IV illustrates the experimental studies with benchmark datasets, experiment settings, and results. Section V summarizes and concludes our paper with future work.

II. RELATED WORKS

Traditional approaches to MWP considers the rule-based matching method [5]. However, the method can solve the problems analyzed by several simple patterns. They cannot extend to the natural language problem sentences whose structure is complex. Several studies [11], [12] use logical expressions to match mathematical problem sentences. They improve the prior work by adding reasoning ability to infer

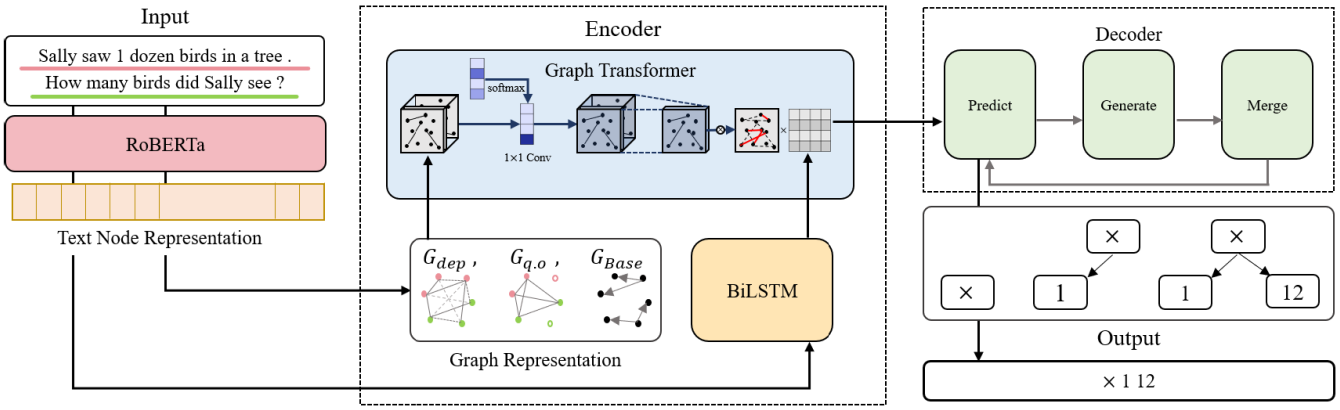


FIGURE 2. Overview of our proposed model.

the meaning of the word. However, they are only feasible to small-sized datasets.

Recent studies have adopted the neural models and proposed benchmark datasets [13], [14], [15]. These approaches use encoder-decoder models whose input is a sequence of math words, and the output is a sequence of operators and numbers. These models exploit the merits of deep learners who can learn the latent features residing in the input sequences. The first neural method, deep neural solver [2], uses the sequence-to-sequence architecture. The work, Seq2SeqET [16], transforms the quantity words into a generalized template. For example, the template equation $X = n_1 + n_2$ represents the given equation $X = 2 + 3$. This method can identify operators whose processing orders are the same. An expression tree can be constructed based on the identification to merge the template equations. The work, StackDecoder [17], extracts the meaning of the quantities expressed in the input sentences while encoding. For decoding, they use a stack structure to contain the meaning of the operands. Most recent approaches exploit the techniques of ML, such as attention mechanism [18] and Transformer [8]. The works [19], [20] use the Transformer to generate the expression of infix, prefix, and postfix notations. Expression-Pointer Transformer (EPT) [21] adds the expression fragmentation and the operand context separation to improve the transformer-based approach. Some work [22] uses the BERT encoder [6] to benefit from the power of the state-of-the-art language model to learn the meaning of natural sentences. However, they lack the structural understanding of MWPs, which are the relationships between the numbers of the words.

Sequence-to-Tree (Seq2tree) [23] proposed a tree-structured equation decoding. The difference from the previous work [17] is that they generate an expression tree whose root node has the operator rather than generating the expression sequentially. The most recent work, Graph-to-tree [3], which adopted Graph Convolution Network [24], to integrate the quantities and the adjacent words as a uniform representation, which is a graph. The state-of-the-art approach [25] defines a math word problem as a relation extraction problem. It models a deductive reasoning process to classify relations. Iterative operations to generate intermediate operation results are proposed. Then, they refine the intermediates until they obtain accurate solutions. This work is closely related to ours because they focus on the explicit reasoning process. The main difference between the work [25] and ours is that it concentrates on transforming the deductive reasoning process to solve MWPs. However, ours focuses on representing and merging a variety of mathematical knowledge. The other recent work, Graph-to-tree [3] is different from ours in that we replace their graph convolution network with the graph transformer network. Also, our method is extensible with the input graph representations.

III. METHOD

We represent the text of the math word problem as P, where P is a sequence of word tokens and numeric values. The problem $P = \{p_1, p_2, \cdots, p_n\}$, is divided into Body and Question as follows: Body $= \{p_1, p_2, \cdots, p_k\}$ and Question $= \{p_{k+1}, p_{k+2}, \cdots, p_n\}$. For example, given problem P, "Sally saw 1 dozen birds in a tree. How many birds did Sally see?". The body part is "Sally saw 1 dozen birds in a tree." and the question part is "How many birds did Sally see?". p_1 denotes the word 'Sally' in the given problem and p_5 denotes the word 'birds' in the body part. Also $p_1 1$ denotes the word 'birds' but it is from the question part. Moreover, $p_1 3$ also denote 'Sally', which is in the question part. There are three types of words: number N_p, word W_p and constant W_{con}. The number N_p is the set of quantities in the problem. The word W_p is the word that occurred in the problem. The constant, for example, dozen is the unit or predefined constants represented by numbers in the problem. Lastly, the operator W_{op}, for example, $\times$ is a mathematical operator in a set of operators $\{+, -, \div, \times\}$. For example, n_{p1} represents the number 3, w_{p1} represents the word 'Sally' and w_{con1} represents the constant word 'dozen,' where $N_p = \{n_{p1}, n_{p2}, \cdots, n_{pl}\}$, $W_p = \{w_{p1}, w_{p2}, \cdots, w_{pm}\}$, and $W_{con} = \{w_{con1}, w_{con2}, \cdots, w_{conn}\}$ are given. In this formulation, w_{p11} also represents the word

'Sally'. Two matching word type elements, w_{p1} and w_{p11}, for a single word 'Sally' will be used to identify the associations in the problem. We assume the association is meaningful in understanding a set of related sentences and is constructed by the word co-reference relationships. The body $\{N_p, W_p, W_{con}\}$ and the question = $\{N_p, W_p, W_{con}\}$ are also represented as a set of three typed words. Given a problem, our goal is to map P to a valid and correct mathematical expression E_p (e.g., 1×12) composed of numbers from P and mathematical operators from the set $W_{op} = \{+, -, \div, \times\}$.

The architecture we propose is shown in Figure 2. Our model consists of three parts: text node representation, graph transformer encoder, and tree-based decoder. First, the model encodes the problem text into a vector representation based on a pre-trained language model. Second, the vector representation is converted to token-level representations using a bidirectional LSTM. Moreover, several graphs are constructed to encode the heterogeneous structural information and are integrated using the graph transformer encoder. Finally, the tree-based decoder generates the solution expression tree.

A. TEXT NODE REPRESENTATION

First, the text node representation module transforms the MWP P texts into vector representations. We learn the word-level hidden state representations of the input MWP text using the RoBERTa [26] pre-trained embeddings $E(p_i)$ to initialize the node representations,. We use pre-trained RoBERTa because our preliminary study showed that its overall performance of it is the best. However, it can be easily replaced with other state-of-the-art language models.

B. HETEROGENEOUS GRAPH-BASED ENCODER

Second, we put $E(p_i)$, which is an embedding matrix of a sequence p_i into the BiLSTM neural network. The BiLSTM neural network encodes the tokens represented in $E(p_i)$ and generates a series of hidden states $h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$:

$$\overrightarrow{h}_i = \text{LSTM}(E(p_i), \overrightarrow{h}_{i-1}) \quad (1)$$

$$\overleftarrow{h}_i = \text{LSTM}(E(p_i), \overleftarrow{h}_{i-1}) \quad (2)$$

where the arrow represents the forward direction of each LSTM. The combined hidden states, $h_i = \overrightarrow{h}_i + \overleftarrow{h}_i$, incorporate the contextual information in $E(p_i)$. Thus, the set of hidden states $H = \{h_1, h_2, \cdots, h_N\}$ represents the holistic context of the problem.

The embedding $E(p_i)$ is also converted to graph structures for the Graph Transformer Network [9] (GTN). We adopt the GTN in our encoder to integrate the input graphs. GTN is the state-of-the-art model in the node classification problem for graph data. According to the work [9], GTN can identify hidden relationships between the original graphs. This means that GTN has the advantage of being able to learn latent information behind the MWP compared to the one that was constructed in a heuristic manner [3]. This ability may correspond to human's reasoning process of solving math problems. Moreover, we find that the meta-paths among heterogeneous types of edges overcome the limitation of graph neural networks [27], which assume a fixed and homogeneous graph structure. The GTN in our encoder takes two input representations: the adjacency matrices of multiple graphs and the context vectors from the BiLSTM. Our main idea is that we can find the hidden relationships between quantities and words using GTN.

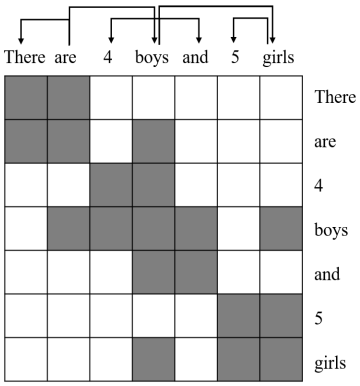


FIGURE 3. Dependency Graph illustrated by an adjacency matrix.

To validate our idea, we propose two graphs, Dependency Graph G_{dep} and Quantity Overlap Graph $G_{q.o}$. These two graphs provide meaningful paths between the nodes connected by the edges of different types. In our encoder, the GTN finds useful meta-paths for enriched expression for generating a final solution. The graph structure G_{Base} in [3] is used as a baseline for our graph representation. In the encoder, the node types of each graph are denoted as $\{E(N_p), E(W_p), E(W_{con})\}$, and the edges types are denoted as $\{A_1, A_2, A_3\}$:

- A_1 : Lexical dependency between words
- A_2 : Words that appear in common in the problem body and the question
- A_3 : Quantity comparison (used by [3])

In this encoder setting, additional edge types can be easily adopted as A_i. The intuitions of the proposed graphs and their edges are described as follows.

1) DEPENDENCY GRAPH

First, we consider the lexical dependency structure between words in the problem. In some cases, the words that are far apart would play an important role in understanding the problem. For example, in Figure 1, we can find the verb 'got' has an association with the subject 'Sam', at the same time, has an association with the quantities, ''32 chewing gums and 15 chocolate bars''. Similarly 'Her sister' is associated with 23 gums and 16 bars. To maintain such a relationship between tokens for MWP, dependency parsing [28] is one of the effective methods in NLP. We adopt the well-known dependency parser [10] to capture the association of distant words and quantities based on lexical dependency. This idea resolves the problem of the prior work [3], which may ignore the relationships between distant tokens that are not adjacent to quantities.

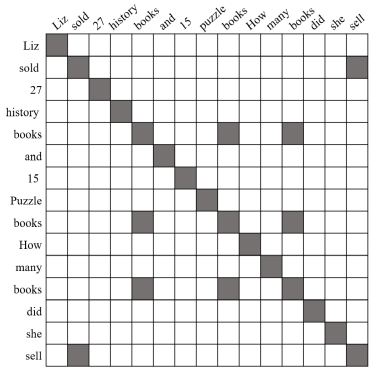


FIGURE 4. Question overlap Graph illustrated by an adjacency matrix.

We introduce Dependency Graph with edges that represent the lexical dependency between nodes (Figure 3). Once a dependency parser determines the dependency structure between token embedding $E(p_i)$, an adjacency matrix A_1 is constructed. Based on the parser, un-directed edges, and their weights are assigned to a matrix, which is a graph representation. We ignore the direction of the edge for simplicity. According to [29], dependency structure forms an important role in converting text into logical formulas. We presume the enhancement of the logical formulations into a training model could affect the final performance of solving MWP.

Figure 3 shows an example of the adjacency matrix to illustrate the dependency relationship between words in the sentence "There are 4 boys and 5 girls". We used the popular NLP toolkit, SpaCy, for dependency parsing. In this study, the parser assigns one for each cell in the matrix if there is a dependency relationship between two words; otherwise, it assigns zero for the cell.

2) QUESTION OVERLAP GRAPH

Second, we try to emphasize the question part of the problem. We conjecture that the previous works, for example, [3], consider adjacency relationships without distinction between the body and the question part. We believe that the words that occurred in the question part are more critical than those in the body part because the question part is comprised of keywords to get a final solution.

An MWP is separated into two parts, the body part and the question part, yet common question words exist. We introduce a co-occurrence graph, called the Question Overlap Graph, with the un-directed edge representing the existence of tokens that occurred in the body part and the question part simultaneously.

Figure 4 shows an example of the adjacent matrix to indicate the overlapping words between the body part, "Liz sold 27 history books and 15 puzzle books", and the question part, "How many books did she sell". In this study, we assign one for each cell in the matrix if common words exist along with the rows and the column of the matrix; otherwise, we assign zero for the cell.

Using the various graph representations with heterogeneous types of edges $\{A_1, A_2, A_3\}$ described above, GTN predicts new edges to generate a new adjacency matrix A_{new}. Then, the encoder generates an output Z obtained by performing a graph convolution operation between A_{new} and H. Followings are the chain of operations processed in the encoder.

$$\mathrm{Z} = \mathrm{GraphConvolution}(\mathrm{GTN}(A_1, A_2, \cdots, A_k), H) \quad (3)$$

where

$$\mathrm{GraphConvolution}(A_k, X) = \mathrm{relu}(A_k X^T W_k) \quad (4)$$

$$\begin{aligned}\mathrm{GTN}(\mathbb{A}) &= \mathrm{F}(\mathbb{A}; W_\phi) \\ &= \phi(\mathbb{A}; \mathrm{softmax}(W_\phi))\end{aligned}$$

$$\mathbb{A} = \{A_1, A_2, \cdots, A_k\} \quad (5)$$

where $\phi(\cdot)$ is the 1×1 convolution layer and $W_\phi \in R^{1\times1\times K}$ is the parameter of ϕ. We use Dependency graph A_1, Question overlap graph A_2, and adjacency matrix used by baseline A_3. We apply Layer normalization and Residual connection to Z to produce the node representation $\bar{Z}$.

$$\hat{Z} = Z + \mathrm{LayerNorm}(Z) \quad (6)$$

$$\bar{Z} = \hat{Z} + \mathrm{LayerNorm}(\mathrm{FFN}(\hat{Z})) \quad (7)$$

where

$$\mathrm{LayerNorm}(x) = \frac{\gamma}{\sigma_x} \cdot (x - \mu_x) + \beta \quad (8)$$

$$\mu_x = \mathrm{E}(x), \quad \sigma_x = (\mathrm{E}(x - \mu_x)^2)^{1/2} \quad (9)$$

$$\mathrm{FFN}(x) = \max(0, x \cdot W_{f_1} + b_{f_1}) \cdot W_{f_2} + b_{f_2} \quad (10)$$

where γ, and β are trainable parameters, W_{f_1} and W_{f_2} are trainable matrices.

We apply the element-wise min pooling operation on all the nodes to learn the global node representation. Next, the global feature is input to a fully connected neural network(FC) to generate the final output z_g of the encoder.

$$z_g = \mathrm{FC}(\mathrm{MinPool}(\bar{Z}) \quad (11)$$

From the output of the encoder, $\bar{Z}$ is used as a node representation of quantities, entries, and relationships. z_g is used as a global context representation for the decoder.

C. TREE-BASED DECODER

Finally, the mathematical equation is generated at the tree-based decoder. We model the tree-based decoder inspired by the Graph-to-tree model [3] to accept the representation output from the encoder module. The decoder takes the integrated context of the problem to generate the sub-expressions sequentially for a final equation. We set the output form of the decoder as a binary tree whose leaf nodes are the numbers and the constants, and non-leaf nodes are the operators. The equation consists of constants, numbers, and operators: $E_p = \{ N_{con}, N_p, W_{op} \}$. We use the pre-order traversal to generate the expression tree in the form of prefix representation since we confirmed that the location of the operator affects the accuracy of generated expressions [19].

1) TREE INITIALIZATION

First, the root node q_{root} is initialized as the output z_g of the graph encoder. Given a problem P, each word y is defined as three types of token embedding as follows.

$$E(y|P) = \begin{cases} \mathrm{M}_{op}(y), & \text{if } y \in W_{op} \\ \mathrm{M}_{con}(y), & \text{if } y \in N_{con} \\ \bar{z}^{p}_{loc(y,P)}, & \text{if } y \in N_P \end{cases} \tag{12}$$

where $E(\cdot)$ is the embedding matrix, $\bar{z}^p$ is the hidden state from the output of the graph encoder, and $loc(y, P)$ is the location index of the word y in the corresponding Problem P. For each word in the vocabulary: $W^{Equ} = \{W_{op}, N_{con}, N_P\}$, embedding matrices M_{op} and M_{con} are trained independently.

Then, the context vector c is initialized by the initial node q_{root}

$$c = \sum_s a_s \bar{Z} \tag{13}$$

where

$$a_s = \frac{\exp(\text{score}(q, \bar{Z}))}{\sum_i \exp(\text{score}(q, \bar{Z}))} \tag{14}$$

and

$$\text{score}(q, \bar{Z}) = v_a^T \tanh(W_a[q, \bar{Z}]) \tag{15}$$

where v_a and W_a are trainable parameters, and $[\cdot, \cdot]$ denotes concatenation operations. The $\mathrm{s}(\cdot)$ generates an output token y given the target word set W^{Equ}.

$$\mathrm{s}(y|q, c, P) = w_n^T \tanh(W_s[q, c, E(y|P)]) \tag{16}$$

where w_n is a trainable vector, W_s is a trainable matrix and $E(y|P)$ is the token embedding of y in Equation (10).

We calculate the probability of the output word y by normalizing the $\mathrm{s}(y|q,c,P)$ using softmax.

$$\text{prob}(y|q, c, P) = \frac{\exp(\mathrm{s}(y|q, c, P)}{\sum_i \exp(\mathrm{s}(y_i|q, c, P)} \tag{17}$$

We can estimate the output token $\hat{y}$ with the highest probability as follows.

$$\hat{y} = \underset{y \in W^{Equ}}{\text{argmax}}\, \text{prob}(y|q, c, P) \tag{18}$$

With the predicted output token $\hat{y}$, we determine whether to generate a sub-tree or terminate. If $\hat{y}$ is a constant or quantity, the decoder stops the generation and decodes it as a leaf node. Otherwise, the decoder proceeds to generate the sub-tree.

2) SUB-TREE GENERATION

We adopt the pre-order traversal to construct the expression tree. In this traversal manner, all the left branches are generated before the right branches. The left child node q_l is constructed as followings:

$$o_l = \sigma(W_{ol}[q, c, E(\hat{y}|P)]) \tag{19}$$
$$C_l = \tanh(W_{cl}[q, c, E(\hat{y}|P)]) \tag{20}$$
$$h_l = o_l \odot C_l \tag{21}$$

where W_{ol} and W_{cl} are trainable matrices, and h_l is defined as the hidden state passed by the parent node to the left child node.

$$g_l = \sigma(W_{gl} h_l) \tag{22}$$
$$Q_{le} = \tanh(W_{le} h_l) \tag{23}$$
$$q_l = g_l \odot Q_{le} \tag{24}$$

where W_{g_l} and W_{le} are trainable matrices.

After generating all the left branches of the sub-trees, the right branches are generated in a similar manner. The difference from generating a left child node is that it utilizes the generated left sub-tree t_l.

$$o_r = \sigma(W_{or}[q, c, E(\hat{y}|P)]) \tag{25}$$
$$C_r = \tanh(W_{cr}[q, c, E(\hat{y}|P)]) \tag{26}$$
$$h_r = o_r \odot C_r \tag{27}$$

where W_{or} and W_{cr} are trainable matrices, and h_r is defined as the hidden state passed by the parent node to the right child node.

$$g_r = \sigma(W_{gr}[h_r, t_l]) \tag{28}$$
$$Q_{re} = \tanh(W_{re}[h_r, t_l]) \tag{29}$$
$$q_r = g_r \odot Q_{re} \tag{30}$$

where W_{g_r} and W_{re} are trainable matrices.

Generated the left and right sub-trees are combined to construct t as follows:

$$t = \begin{cases} \text{comb}(t_l, t_r, \hat{y}) & \text{if } \hat{y} \in W_{op} \\ E(\hat{y}|P), & \text{if } \hat{y} \in N_p \cup N_{con} \end{cases} \tag{31}$$

$$\text{comb}(t_l, t_r, \hat{y}) = g_t \odot C_t \tag{32}$$
$$g_t = \sigma(W_{gt}[t_l, t_r, E(\hat{y}|P)]) \tag{33}$$
$$C_t = \tanh(W_{ct}[t_l, t_r, E(\hat{y}|P)]) \tag{34}$$

Suppose the predicted token value $\hat{y}$ is an operator. In that case, the decoder generates the operator and proceeds the prediction for the operands in the form of two sub-trees, t_l and t_r based on the combination $\text{comb}(t_l, t_r, \hat{y})$, where W_{g_t} and W_{ct} are trainable matrices. If the predicted token value $\hat{y}$ is a number or a constant, the decoder generates the operand in the form of a value, $E(\hat{y}|P)$.

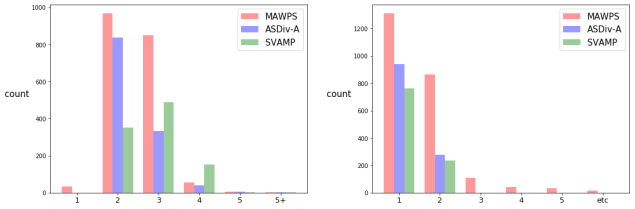
If the output meets the values N_p, N_{con}, and all the sub-trees are generated, the tree-based decoder outputs the equation generated so far.

D. TRAINING

Our objective is to minimize the negative log-likelihood of the set of MWPs. The loss function we used for model learning is as follows.

$$L(E_p|P) = \sum_n (-\log p(E_p|P)) \tag{35}$$

$$p(E_p|P) = \prod_{i=1} \text{prob}(y_i|q_i, c_i, P) \tag{36}$$



(a) The number of quantities (b) The number of operators

FIGURE 5. Statistics of three MWP datasets.

where n denotes the length of E_p, and q_i and c_i are the tree node vector, and its context vector at the i-th token, the probabilities prob(·|·) are calculated by Equation (15). We used Adam optimizer to train the model with the fixed learning rate 1e-3. The weights are initialized using a normal distribution with zero mean and one standard deviation. The weights of GTN layer are initialized with zero mean and 0.01 standard deviation.

IV. EXPERIMENTS

A. DATASETS

To evaluate our proposed method, we use three benchmark datasets: MAWPS [14], ASDiv-A [15], and SVAMP [13]. MAWPS is a dataset that contains simple MWPs that can be solved by a linear equation with a single variable. It has a total of 2373 problems with 1311 single operator problems and 1062 multiple operator problems. This dataset is constructed by merging several public datasets, such as AI2 [30], IL [31], and SingleEQ [32]. ASDiv(Academia Sinica Various MWP) is a dataset of simple MWPs similar to MAWPS. Moreover, it contains useful information about the problem difficulty and the problem types. We use the type of Arithmetic problem, ASDiv-A, that includes 1218 problems. SVAMP is a special subset of ASDiv-A to sample 1000 problems with a variation. The problems in the datasets we used are described in English only.

TABLE 2. Statistics of Dataset.

Dataset	Problems	Words	Avg Ops	CLD
MAWPS	2373	2452	1.78	0.26
ASDiv-A	1218	2877	1.23	0.50
SVAMP	1000	833	1.24	0.22

Table 2 shows the statistics of three different datasets. The number of problems and unique words, the average number of operators, and the Corpus Lexicon Diversity (CLD) are illustrated. Based on [15], the CLD metric shows that a dataset with higher CLD has more diverse tokens to express the corpus. For MAWPS, which has the largest number of questions and unique words, we observed that the sentence structures are similar, yet only the objects and quantities are distinct. For ASDiv-A, which is about half the size of MAWPS, has 2877 unique words, and the CLD of it is the highest. We can consider that ASDiv-A contains more diverse tokens, which can form various problem statements than other datasets. The SVAMP dataset is created by applying certain variations to a set of seed examples sampled from the ASDiv-A dataset. The problem with this dataset is that there exist many augmentation data derived from a single sample. As a result, the CLD is low because the vocabulary is too small; also the sentences have a few variations with the same sentence structure.

Figure 5a and 5b show that MAWPS has a relatively large number of problems with two or three quantities and one or two operators, and SVAMP has a relatively large number of problems with four quantities.

B. EXPERIMENTAL SETTINGS

The experiments were performed on the Intel i7 3.2 GHz workstation with 256GB of RAM. Furthermore, the general graphics processing unit, RTX 3090 was used to train our models and other baseline models. Every model is implemented using the libraries, such as Python 3.6.9, Pytorch 1.7.1, Pandas, Numpy, Scikit-Learn, Huggingface Transformers, NLTK, and SpaCy, on the Ubuntu 18.04 LTS operation system environment. We optimized the hyper-parameters of our model of input embedding size of 768, the number of units in a hidden layer of 384, and two layers in the encoder with an RTX3090 24GB GPU. We trained our model for 50 epochs with hyper-parameters (learning rate: 1e-3, batch size: 8, weight decay: 1e-5, beam size:5, dropout: 0.5).

To test the model performance, we compare the Equation Accuracy and the Answer Accuracy of our model and the baseline models. We measure the metric using 5-fold cross-validation. For comparison, the recent studies [3], [16], [19], [23] are presented as baseline methods denoted as, Sequence-to-Sequence, Transformer, Seq2tree, and Graph-to-tree, respectively.

Our approach is presented with four variants, Base(G_{dep}), Base($G_{q.o}$), Base(GTN), and HGTE2Tree to illustrate the increasing performance by adding heterogeneous graphs and GTN to a single end-to-end model. The followings summarize the baseline methods and our model variants. Our code is at "https://github.com/soyunshin/HGTE2Tree".

1) Sequence-to-Sequence model is denoted as seq2seq. It is the encoder-decoder architecture using a Bi-directional GRU for encoding and a Uni-directional GRU for decoding.
2) Vanilla Transformer in the Pytorch library is denoted as Transformer. It is constructed as a single layer of encoder and decoder with four heads.
3) Sequence-to-Tree model is denoted as Seq2tree. It uses a Bi-directional GRU to represent the goal vector for tree generation.
4) Graph-to-tree model, denoted as Graph2tree(Base), transforms the problem inputs into a word embedding using a Bi-directional GRU. The Quantity Cell Graph is constructed with the most adjacent two words as input graph representation. The decoder generates the final tree of equations whose depth is only two.

5) Baseline + G_{dep} is our approach to introduce the dependency structure to adopt the heterogeneity into the graph representation, denoted as Base+G_{dep}. It extends the Graph2tree model by adopting the Dependency Graph G_{dep}.
6) Baseline + $G_{q.o}$ is our approach to introduce the word overlaps, denoted as Base+$G_{q.o}$. It extends the Graph2tree model by adopting the Question overlap Graph $G_{q.o}$.
7) Baseline + GTN is our approach to integrate the heterogeneous graphs, denoted as Base+GTN. It extends the encoder of Graph2tree(Base) by adopting GTN. This approach integrates the Quantity Cell Graph and the Quantity Comparison Graph into a single graph.
8) Heterogeneous Graph Transformer Encoder-to-Tree is our proposed model, denoted as HGTE2Tree. It transforms the input with integrated two novel graphs, Dependency Graph G_{dep} and the Question Overlap Graph $G_{q.o}$, and BiLSTM for the GTN in the encoder. They are transformed into a single graph representation for the tree-based decoder.

TABLE 3. Overall performance: equation accuracy and answer accuracy.

Model	Accuracy	MAWPS	ASDiv-A	SVAMP
seq2seq	Equation	84.11	71.43	60.14
	Answer	86.56	75.49	62.22
Transformer	Equation	82.67	70.85	61.07
	Answer	85.94	73.94	63.37
Seq2tree	Equation	86.46	76.40	59.41
	Answer	87.50	80.05	62.69
Base(Graph2tree)	Equation	87.40	77.94	60.74
	Answer	88.40	82.15	64.27
Base + G_{dep}	Equation	87.24	78.57	62.61
	Answer	88.54	83.19	66.99
Base + $G_{q.o}$	Equation	88.02	78.15	61.22
	Answer	88.54	83.19	65.15
Base + GTN	Equation	**88.81**	**81.09**	60.70
	Answer	89.10	82.77	64.94
HGTE2Tree	Equation	88.54	79.83	**63.05**
	Answer	**89.58**	**84.45**	**67.20**

C. OVERALL PERFORMANCE

Table 3 shows the overall performance of MAWPS, ASDiv-A, and SVAMP. HGTE2Tree illustrates better results than the baseline for three datasets. This result explains that the input representation of heterogeneous edges contributes to the accuracy of reasoning a final equation and solution given MWPs. With the results of two models, Base+G_{dep} and Base+$G_{q.o}$, we find that additional information on relationships between words and numbers in the problem is helpful in improving performance. Moreover, the integration with heterogeneous graph representations using GTN, Base+GTN, yields a large portion of the improvement in Equation accuracy compared to the baselines. Also, HGTE2Tree shows state-of-the-art performance in Answer accuracy.

For Base+G_{dep}, which adds the Dependency Graph to the baseline, the accuracy of equation generation on the MAWPS dataset is reduced compared to the baseline. Still, the accuracy of answer generation is improved. By adding the dependency graph, the relationship between quantities and distant words may be considered, resulting in an error in the order of equations. Still, even if the order of equations is predicted differently, the accuracy of the correct answers calculated increases by 0.54%. For the ASDiv-A dataset, the equation accuracy and the answer generation accuracy are improved by about 1%. These results illustrate that the Dependency Graph can efficiently incorporate far-flung word information involved in equations and correct answers in the problem of ASDiv-A, which is relatively complex compared to other datasets. Even in the case of SVAMP datasets, the accuracy of equation and answer generation has been improved.

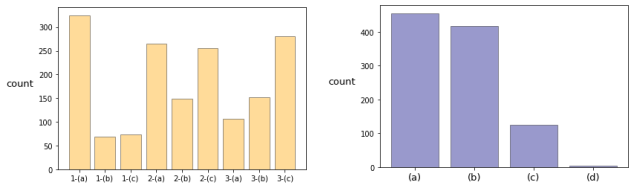
For Base+$G_{q.o}$ with Question overlap Graph added to the baseline, the generation accuracy for the equation and the answer is improved for all datasets. By adding a Question overlap Graph, it can be inferred that the process of finding the necessary information in the body part of the question in the question part of the problem and inferring it by an equation has a good effect on performance.

For Base + GTN, the accuracy of equation and answer generation on the MAWPS dataset is improved. Adopting the GTN to our encoder, the model can find a new meta-relation based on the initial relationship information with the quantity and the word has a positive effect on performance. In addition, the generation accuracy of equations and answers on the ASDiv-A dataset is improved. However, for SVAMP datasets, the accuracy of equation generation decreased by 0.04%, but the accuracy of correct answer generation improved. In the case of various variations of existing problems, such as SVAMP datasets, it can be considered that the method of inferring information through meta-relation generated through learning data negatively affects the accuracy of equation generation.

For our proposed model HGTE2Tree, the equation and answer generation accuracy on the MAWPS dataset is improved by approximately 1%. On the ASDiv-A dataset, the equation and answer generation accuracy is improved by more than about 2%. The equation and answer generation accuracy on the SVAMP dataset was improved by more than about 3%. It can be observed that the use of GTN has a positive effect on performance improvement so that the information of the graph representing each structural information is not excessively learned in the process of generating a new relationship structure by adding the dependency graph and question overlap graph.

D. VARIATIONS IN MWP

Based on the benchmark dataset, SVAMP [13], we conducted an experiment to analyze the reasoning ability of the models in terms of three metrics: Question Sensitivity, Reasoning Ability, and Structural Invariance. The followings are the candidate variations for each metric.



(a) The number of problems for each variation (b) The number of Variations in each problem

FIGURE 6. Statistics of SVAMP variation data.

1) Question Sensitivity is measured while changing the question part. We applied the following variations to the models with the problem body part fixed.
   a) Same Object, Different Structure - changed the structure of the question while the objects in the question are fixed.
   b) Different Object, Same Structure - changed the objects in the question while the question structure is fixed.
   c) Different Object, Different Structure - changed the objects and the question structure.
2) Reasoning Ability is measured while adding some changes to the problem. We evaluated the changing performance according to the following variations.
   a) Add relevant information - added relevant information that could affect the output equation quality into the problem.
   b) Change Information - changed the object information in the problem.
   c) Invert Operation - changed the quantity of objects in the problem while the question was fixed.
3) Structural Invariance is a metric to evaluate how the model performance could be affected if some superficial changes occurred in the problem. The followings are the changes we performed.
   a) Change order of objects - changed the order of the quantities and the objects in the problem.
   b) Change order of Phrases - changed the order of the quantities and the phrases in the body part.
   c) Add irrelevant information - added irrelevant information to the problem.

We conducted this experiment to evaluate the performance of our model using SVAMP datasets. We assess the ability to reason the equations and answers concerning each variant category. Each problem can be classified into at least one or more categories (maximum of four). In Figure 6a, the number of problems belonging to 9 variants described above is illustrated. Figure 6b shows the number of variant properties in each problem. That is, (a) means having one variant property, (b) means having two variant properties at the same time, (c) means having three variant properties at the same time, and (d) means having four variant properties at the same time. We measure the metric using 5-fold cross-validation.

Table 4 shows that our model, HGTE2Tree outperforms the baseline, Graph2Tree, for the several variations we applied. In most cases, our model has improved formulation and corrective inference performance. Except for the one in which we changed the objects with the same structure in the question, the accuracy of our model did not deteriorate with various problem changes. However, in the case of Question Sensitivity's Different Object and Same Structure, it was confirmed that the baseline model's performance was about 1.3% higher for Equation generation and about 3.8% higher for Answer generation. The Graph-to-tree model predicted one equation and two answers more accurately than the proposed model. This is the case when an object that does not exist in the training data appears in a new question, and our proposed HGTE2Tree model recognizes objects that are semantically associated with Quantity as Quantity-linked information, so if the object changes and the structure remains the same, it is observed that the performance is relatively low.

TABLE 4. SVAMP variation.

CATEGORY	VARIATION	Acc	Graph2Tree	HGTE2Tree
Question Sensitivity	Same Obj, Diff Struct	Equ	54.64	**59.70**
		Ans	57.66	**61.17**
	Diff Obj, Same Struct	Equ	**55.80**	54.55
		Ans	**62.47**	58.72
	Diff Obj, Diff Struct	Equ	59.44	**60.56**
		Ans	64.72	**65.83**
Reasoning Ability	Add Rel Info	Equ	56.75	**58.81**
		Ans	63.49	**64.18**
	Change Info	Equ	48.69	**50.47**
		Ans	54.22	**55.44**
	Invert Operation	Equ	80.63	**82.27**
		Ans	83.67	**84.85**
Structural Invariance	Change order of Obj	Equ	59.25	**60.30**
		Ans	64.39	**65.44**
	Change order of Phrases	Equ	70.12	**72.08**
		Ans	75.10	**76.53**
	Add Irrel Info	Equ	53.43	**55.89**
		Ans	55.08	**58.60**

V. QUALITATIVE EVALUATION RESULTS

We performed the case study to discuss the qualitative evaluation results concerning the SVAMP variation. The results are shown in Figure 7.

The first case is the problem of Question Sensitivity. The problem requires the number of 'files' remaining after 'delete'. It is necessary to distinguish between the quantity related to 'files' and quantity related to 'apps'. Our model successfully finds the quantities associated with 'files' and deduces the equation. However, the baseline fails to recognize the order of appearance of 'files' and 'apps' has changed at the beginning and the end of the sentence, and the quantity 17 of 'apps' are included in the equation.

The second case is the problem of Reasoning Ability. Different operators are observed in generating equations in our

Variation : Question Sensitivity			
Question	Dave had 24 files and 13 apps on his phone . After deleting some apps and files he had 17 apps and 21 files left . How many files did he delete ?		
Baseline	24 – 17 = 7	Our model	24 – 21 = 3
Variation : Reasoning Ability			
Question	Marco and his dad went strawberry picking . Together they collected strawberries that weighed 36 pounds . On the way back marco ' dad lost 8 pounds of strawberries . Marco 's strawberries now weighed 12 pounds . How much did his dad 's strawberries weigh now ?		
Baseline	36 – 12 + 8 = 32	Our model	36 – 12 – 8 = 16
Variation : Structural Invariance			
Question	Rebecca wants to split a collection of eggs into groups of 6 . Rebecca has 18 eggs 72 bananas and 66 marbles . How many groups will be created ?		
Baseline	72 ÷ 6 = 12	Our model	18 ÷ 6 = 3

FIGURE 7. Case Studies on Variations in SVAMP.

model and baseline. It was observed that the baseline added 8 to create an equation. The 8 is the amount that 'his dad' lost. However, our model successfully deduced the quantity of 'his dad.'

Finally, the third case is the problem of Structural Invariant. It is necessary to identify the quantity associated with 'eggs' to solve the problem. Our model successfully finds a quantity 18 related to 'eggs' and deduces the equation. However, for the baseline, the equation contains the quantity 72 of 'bananas' located close to 'eggs' instead of the quantity 18 of 'eggs'.

To provide various types of structural information in a uniform way, we propose a graph transformer encoder to integrate heterogeneous graphs of various input representations.

VI. CONCLUSION

This paper introduces a novel method to integrate heterogeneous graphs into a single uniform graph representation using a graph transformer encoder for solving MWP. We proposed two graphs, Dependency Graph, and Question Overlap Graph, to extract the structural information for reasoning a mathematical equation from the problem sentences. We discussed that adopting the graph transformer network helps reconstruct the latent structures behind the problem sentences and decode them in tree-structured expressions. Our evaluation indicates that our method outperforms the baselines regarding equation accuracy, answer accuracy, question sensitivity, reasoning ability, and structural invariance.

The limitation of this paper is that we have only proposed two types of heterogeneous graphs. They may be considered the same type of graphs because they can be represented as an adjacent matrix. If we would elaborate our work more concretely in a unified framework for various input graphs, more numbers of graph edges and nodes to generalize our claims. The graph-based models have the potential to gather a variety of mathematical components into a single unified framework. The second limitation of this work is that it only considers the arithmetic operators, and a limited set of experiments has been presented. Although arithmetic problems are in the scope of this work, we believe our graph transformer encoder would be optimized for the equation set problems.

As a future work, we will try to apply our method to other types of MWP, such as equation set problems. Moreover, additional graphs, such as Abstract Meaning Representation graph [33], are to be considered for rich information. As a future work of model enhancement, we plan to improve our model performance by relying on large-scale language models, such as PaLM [34] or GPT3 [35]. We believe adopting the pre-trained models would broaden the range of reasoning tasks at the math word problem representation level.

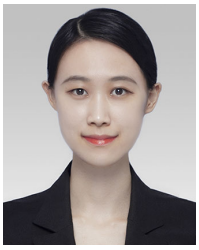
ACKNOWLEDGMENT

(Jaehui Park and Moonwook Ryu contributed equally to this work.)

REFERENCES

[1] D. Zhang, L. Wang, L. Zhang, B. T. Dai, and H. T. Shen, "The gap of semantic parsing: A survey on automatic math word problem solvers," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2287–2305, Sep. 2020.
[2] Y. Wang, X. Liu, and S. Shi, "Deep neural solver for math word problems," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 845–854.
[3] 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 *Proc. Assoc. Comput. Linguistics*, 2020, pp. 3928–3937.
[4] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, "Learning to automatically solve algebra word problems," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics (Long Papers)*, vol. 1, 2014, pp. 271–281.
[5] C. R. Fletcher, "Understanding and solving arithmetic word problems: A computer simulation," *Behav. Res. Methods, Instrum., Comput.*, vol. 17, no. 5, pp. 565–571, Sep. 1985.
[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[7] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018, *arXiv:1802.05365*.
[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 5998–6008.
[9] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 11960–11970.
[10] B. Srinivasa-Desikan, *Natural Language Processing and Computational Linguistics: A Practical Guide to Text Analysis With Python, Gensim, Gensim, spaCy, Keras*. Birmingham, U.K.: Packt Publishing Ltd, 2018.
[11] D. Goldwasser and D. Roth, "Learning from natural instructions," *Mach. Learn.*, vol. 94, no. 2, pp. 205–232, Feb. 2014.
[12] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, "Scaling semantic parsers with on-the-fly ontology matching," in *Proc. Conf. empirical methods natural Lang. Process.*, 2013, pp. 1545–1556.
[13] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP models really able to solve simple math word problems?" 2021, *arXiv:2103.07191*.
[14] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "MAWPS: A math word problem repository," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1152–1157.
[15] S.-Y. Miao, C.-C. Liang, and K.-Y. Su, "A diverse corpus for evaluating and developing English math word problem solvers," 2021, *arXiv:2106.15772*.
[16] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to an expression tree," 2018, *arXiv:1811.05632*.

[17] T.-R. Chiang and Y.-N. Chen, "Semantically-aligned equation generation for solving and reasoning math word problems," 2018, *arXiv:1811.00720*.
[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.
[19] K. Griffith and J. Kalita, "Solving arithmetic word problems automatically using transformer and unambiguous representations," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2019, pp. 526–532.
[20] K. Griffith and J. Kalita, "Solving arithmetic word problems with transformers and preprocessing of problem text," 2021, *arXiv:2106.00893*.
[21] B. Kim, K. S. Ki, D. Lee, and G. Gweon, "Point to the expression: Solving algebraic word problems using the expression-pointer transformer model," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 3768–3779.
[22] Z. Liang, J. Zhang, L. Wang, W. Qin, Y. Lan, J. Shao, and X. Zhang, "MWP-BERT: Numeracy-augmented pre-training for math word problem solving," in *Proc. Findings Assoc. Comput. Linguistics, (NAACL)*, 2022, pp. 997–1009.
[23] Z. Xie and S. Sun, "A goal-driven tree-structured neural model for math word problems," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 5299–5305.
[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[25] Z. Jie, J. Li, and W. Lu, "Learning to reason deductively: Math word problem solving as complex relation extraction," 2022, *arXiv:2203.10316*.
[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[27] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2016.
[28] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 740–750.
[29] S. Reddy, O. Täckström, M. Collins, T. Kwiatkowski, D. Das, M. Steedman, and M. Lapata, "Transforming dependency structures to logical forms for semantic parsing," *Trans. Assoc. Comput. Linguistics*, vol. 4, pp. 127–140, Dec. 2016.
[30] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, "Learning to solve arithmetic word problems with verb categorization," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 523–533.
[31] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, "Parsing algebraic word problems into equations," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 585–597, Dec. 2015.
[32] S. Roy and D. Roth, "Solving general arithmetic word problems," 2016, *arXiv:1608.01413*.
[33] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract meaning representation for Sembanking," in *Proc. 7th Linguistic Annotation Workshop Interoperability With Discourse*, 2013, pp. 178–186.
[34] A. Chowdhery, "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.
[35] T. B. Brown, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.



SOYUN SHIN received the B.S. degree in statistics and computer software from Sungshin Women's University, Seoul, South Korea, in 2019, and the M.S. degree in statistics from the University of Seoul, South Korea, in 2022. Her research interests include statistical learning, data mining, and machine learning.

JAEHUI PARK received the B.S. degree in computer science from Korea Advanced Institute of Science and Technology, South Korea, in 2005, and the M.S. and Ph.D. degrees in computer science and engineering from Seoul National University, South Korea, in 2008 and 2012, respectively. From 2012 to 2018, he was a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea. He was an Assistant Professor with the Department of Computer Science and Engineering, Incheon National University, South Korea, from 2018 to 2020. Since 2020, he has been an Assistant Professor with the Department of Statistics, University of Seoul, South Korea. His research interests include database applications, big data processing, machine learning, and statistical analysis.

MOONWOOK RYU received the B.S. degree in electronic and electrical engineering from Sungkyunkwan University, South Korea, in 2006, and the M.S. degree in electrical and computer engineering in South Korea, in 2008. Since 2013, he has been a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea. His research interests include human behavior recognition, physiological measurement, and machine learning.