Improving a Graph-to-Tree Model for Solving Math Word Problems

Hyunju Kim*†, Junwon Hwang*‡, Taewoo Yoo*§, and Yun-Gyung Cheong*¶
*SungKyunKwan University (SKKU), Suwon, South Korea
†ESTsoft, Seoul, South Korea
†julia981028@gmail.com, ‡nuclear1221@gmail.com, §woo990307@naver.com, ¶aimecca@skku.edu

Abstract—**In the area of Math Word Problem (MWP), various methods based on deep learning technology have been actively researched. Graph-to-Tree (Graph2Tree) is one of those methods which uses a graph-based encoder and a tree-based decoder to understand the word problem and to generate a valid equation. This method is proven to be well-performed by achieving state-of-the-art on several benchmarks. However, on the benchmark of SVAMP, recent methods including Sequence-to-Sequence (Seq2Seq), Goal-driven Tree-Structured MWP Solver (GTS), and Graph2Tree performs poorly, unable to cope with several variation types that requires natural language comprehension capability. In this paper, we propose an improved version of Graph2Tree which considers the characteristics of natural language to understand the word problems. On top of the original Graph2Tree model, we additionally build *Dependency Graph* and enhance the *Quantity Cell Graph* to *Softly Expanded Quantity Cell Graph*. This helps a graph-based encoder to capture the relationship among words. Also, we introduce *question embedding* for the tree-based decoder to generate equation based on the question given as input. We conduct experiments to evaluate our model against the original Graph2Tree model on three available datasets: MAWPS, ASDiv-A, and SVAMP. We also present case studies to qualitatively examine the effectiveness of the methods and showed that our methods have improved the original Graph2Tree model.**

I. Introduction

Math Word Problem (MWP) is a task of deriving mathematical equations and solutions when a question is given as the short text. To solve this task, strong natural language understanding ability, the world knowledge, and domain knowledge are needed. This is because the machine needs to fully understand the information that the question implies and needs to know what that question is asking machine to solve. Also, to derive the equation, machine needs to capture the relation between numbers and natural language and needs to extract relevant information that is needed to derive an equation to obtain the final solution for the given problem.

Up until now, there have been a lot of approaches trying to solve MWPs. Wang et al. [1] applied a vanilla sequence-to-sequence (seq2seq) model which maps a mathematical question to a solution equation. Encountering limitation to express the equation using only sequence, researchers made attempts to use the tree expressions [1], [2] and tree-based decoders [3], [4].

Following Graph Convolutional Network (GCN) [6], graph-based neural networks [8]–[11] showed good performance on various Natural Language Processing (NLP) tasks. For MWP task also, encoder-decoder models with graph network [12] and graph transformer based models [13], [14] have been proposed. Especially, Graph-to-Tree (Graph2Tree) model developed by Zhang et al. [15] combine a graph transformer and a tree-based decoder and achieved the state-of-the-art in various MWP benchmarks.

TABLE I
An example of math word problem

Problem	Body	James want to distribute 190 apples to 4 students equally. When he took out the apples, 126 of them were rotten.
	Question	How many apples can each person have?
Answer	16	
Equation	(190 - 126) / 4	

To evaluate the performance of the methods on MWP, various datasets including Math23K [16], MAWPS [17], and ASDiv [18] are used. The data contain mathematical problems, equations, and solutions (see table I for example). Recently, Patel et al. [19] tested several models including the state-of-the-art model, Graph2Tree, and found out that the previous datasets are not robust enough to serve as the benchmark datasets for evaluating the performance of the MWP task. To solve this problem, they created a new dataset, called SVAMP, by applying some variations over the word problems of one-unknown arithmetic MWP on a datasets of MAWPS and ASDiv-A. The results showed that existing models have much lower performance when tested on SVAMP than when tested on the previous datasets.

Inspired by the work in [19], this paper presents Graph2Tree$^+$, an enhanced version of Graph2Tree considering variation types that SVAMP made on the benchmarks of MAWPS and ASDiv-A as an indicator of improvements. In particular, we design two graphs that can further enrich the information given to the graph-based encoder. The first one is *Dependency Graph* that defines the relationship between words. By letting model to know this relationship, we expect that the model can understand more about the text in MWP with information using characteristics of natural language. Second, we construct *Softly Expanded Quantity Cell Graph*. This graph is an improved version of the previous *Quantity Cell Graph* presented in Graph2Tree, which adds soft connection between quantity and words nearby the word related to

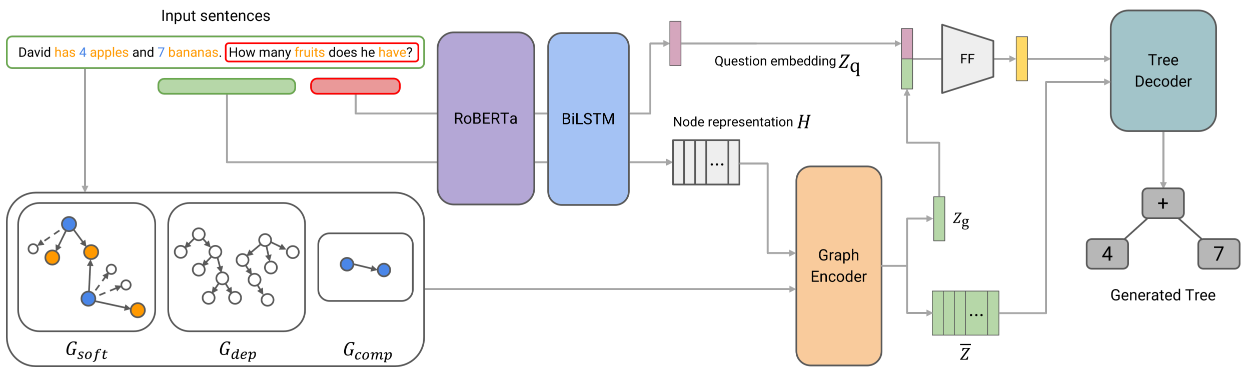

Fig. 1. A whole model structure of Graph2Tree$^+$

quantity. With this, we expect model can use more linguistic information for generating equations. Finally, we introduce question embedding vectors to provide the information of question sentence. By feeding this additional question information to the tree-based decoder we expect the model to generate each tree node considering what exactly the math word problem asks to calculate.

The contributions of this paper are as follows:

- We propose methods of improving graph representation of the math word problem: **Dependency Graph** and **Softly Expanded Quantity Cell Graph**. With these, models can train using linguistic information of math word problem and can enrich the relation between words.
- We employ **question embedding** for the tree-based decoder when generating an equation. With this embedding, the model can generate equation conditioned on the question sentence, making sure that the generated equation is related to what the question is asking.
- We conduct experiments on three available MWP benchmark datasets and prove that our method outperforms the original Graph2Tree model, and are more robust in the case where variations are made onto the word problems.

II. Background

In this section, we first formulate the MWP task. Then, we briefly review the Graph2Tree [15] approach to solve the MWP task and some variations made on the previous datasets to build SVAMP [19] for robust evaluation.

A. Problem Formulation

The Math Word Problem (MWP) task requires machine to derive an equation that can actually be calculated when natural language problem is given. We denote the word problem as P, which is a sequence consisting of word tokens and numeric values. Word tokens are represented as $W_p = \{w_1, w_2, \cdots, w_m\}$, whereas numeric values are denoted as $N_p = \{n_1, n_2, \cdots, n_l\}$. To solve MWP, our goal is to map P to a correct mathematical equation, denoted as E_p, thus estimating a conditional probability of $P(E_p|P)$. Since we focus on arithmetic operations only, E_p contains $O \in \{+, -, \times, \div\}$ and numbers N_p.

B. Graph-to-Tree

To understand and to retrieve the relations in the given problem P, Graph2Tree [15] uses graph-based encoder inspired by the graph transformer model [13]. This model first initialize the input node representation by learning hidden state representations of bidirectional LSTM neural networks. Then, two graphs named *Quantity Cell Graph* and *Quantity Comparison Graph* are created.

First, *Quantity Cell Graph* (G_{qcell}) tries to enrich the representation of the quantities using the information given by the natural language description of the problem. It is constructed by connecting the quantity n_i and its related nodes $w_j \in \{w_{1i}, \cdots w_{qi}\}$, which are extracted by using dependency parsing, constituency parsing, and POS tagging. Secondly, *Quantity Comparison Graph* (G_{qcomp}) is added to give an information about the numerical value of a quantity and the relations between the quantities. Note that a directed edge between n_i to n_j is created only when $n_i > n_j$ is true. Those two graphs are represented as adjacency matrices (A_{qcell} and A_{qcomp}), which are provided along with the initial node embeddings as an input to the graph transformer to construct the entire graph representation z_g.

Now that the model have the knowledge extracted from the given word problem, it uses a tree-based decoder inspired by the Goal-driven Tree Structure (GTS) [4] to generate a valid equation for the given problem. This tree-based decoder builds the root node q_{root} using the graph representation z_g for the tree-based decoder to start with. Then, it generates the left child nodes conditioned on their parent node until the leaf node is produced, then it generates the right nodes. When generating a node, its value is predicted - if the predicted value is an operator, two empty child nodes are created, and if it is a quantity, the node becomes a leaf node.

C. SVAMP

By showing that the performance of various models significantly drops when they apply small variations over an existing

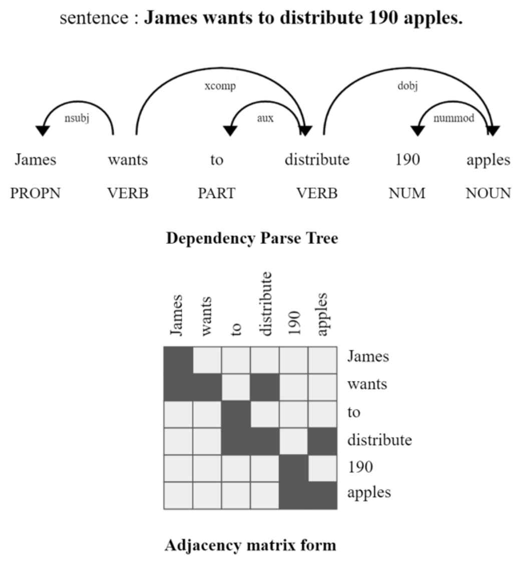

Fig. 2. An example of dependency graph

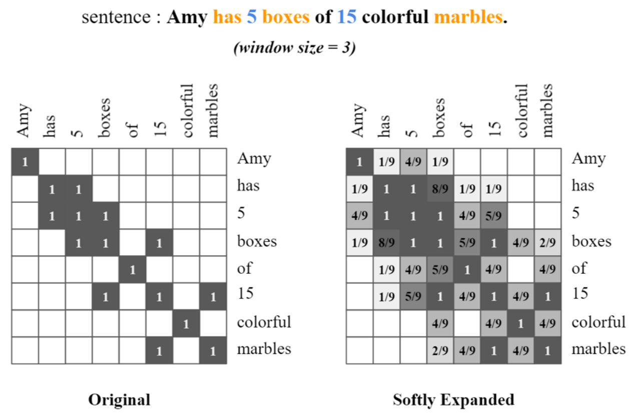

Fig. 3. An example of softly expanded quantity cell graph

benchmarks, researchers of SVAMP suggest that the existing benchmarks of MWP are not robust enough to evaluate model performance. Thus, they built a new benchmark dataset, SVAMP, by applying variations on MAWPS and ASDiv-A, which are English MWP datasets of arithmetic problems.

The variations they made can be categorized into three types based on the ability they can expect for the ideal model: Question Sensitivity, Reasoning Ability, and Structural Invariance. Question Sensitivity changes either object or structure of the question, expecting model to derive the right equation when only the question of the problem is changed. Reasoning Ability adds or change information provided on the problem or invert the operation, expecting model to determine a change in reasoning of problem text. Structural Invariance adds irrelevant information or change order of objects or phrases, expecting model to understand the relationship between information retrieved so that it knows what is the information needed for problem solving.

III. Methodology

In this paper, we propose three methods that can be applied to the original Graph2Tree to enhance their performance. Our model, Graph2Tree$^+$, first encodes MWP text input using RoBERTa and put it into the BiLSTM to get the node representation. Simultaneously, three graphs are constructed: *Quantity Comparison Graph* (G_{qcomp}), *Softly Expanded Quantity Cell Graph* (G_{soft}), and *Dependency Graph* (G_{dep}). Method for constructing *Quantity Comparison Graph* has not been changed from the original Graph2Tree. Each of these graphs enriches the information about relationship between quantities, between quantity and words, and between words. After converting these three graphs as adjacency matrices, graph-based encoder learns entire graph representation z_g using those matrices and learned node representation as an input. Then, we get question embedding z_q and put it to global context z_g to make sure that the model knows what given MWP is asking. Along with z_g, z_q is used for tree-based decoder to generate the target equation that can be calculated for final solution. See Fig. 1 for more understandings.

A. Dependency Tree

The graph-based encoder of the original Graph2Tree model uses *Quantity Cell Graphs* and *Quantity Comparison Graph* as an input. *Quantity Cell Graphs* represent the relation between quantity and words whereas *Quantity Comparison Graph* represents the relation between quantities. However, none of them considers the relationships between words. The word problem P is described in text and it is important to understand the relationship between words when understanding the natural language. When those relationships are well captured by the model, it can relate right information to solve the problem and can perform well regardless of the change in sentence structure or phrases.

We simply construct an additional graph call *Dependency Graph* to represent the word relationships. We first parse the MWP sentence to retrieve the dependency between words. This dependency represents the relationship between the words in P. Then, we convert the parsed tree into a graph by connecting parent node to a child node using a directed edge. The generated graph then transforms to adjacency matrix A_{dep} by setting its value to 1 when two different words are connected or when a word is connected to its own-self. An example is illustrated in Fig. 2, which shows a dependency parse tree and adjacency matrix formed with the sentence of "James wants to distribute 190 apples". For adjacency matrix, black cell indicates the edge value of 1 when light gray cell indicates 0. The diagonal of the adjacency matrix is black since the words are connected to themselves.

B. Softly Expanded Quantity Cell Graph

In the original Graph2Tree model, *Quantity Cell Graph* G_{qcell} was constructed by connecting quantity n_i and related words w_j, setting its value to 1 for its adjacency matrix.

TABLE II
RESULT OF EACH MODEL FOR BENCHMARK DATASETS

	MAWPS	ASDIV	SVAMP
Graph2Tree	88.1	80.0	64.5
Graph2Tree + z_q	88.0	81.3	65.0
Graph2Tree + G_{dep}	89.0	81.2	65.7
Graph2Tree + G_{soft}	88.9	81.2	65.2
Graph2Tree$^+$	**89.4**	**81.4**	**66.6**

TABLE III
RESULT OF GRAPH2TREE MODELS IN VARIATION CATEGORIES

	Graph2Tree	Graph2Tree$^+$
Question Sensitivity	49.8	**57.0**
Reasoning Ability	64.0	**67.8**
Structural Invariance	62.2	**62.6**

This let model to successfully know whether or not quantities and words are related to each other. However, this method assumes that the quantity-related words are correctly chosen. Furthermore, it has an limitation that it is hard to use the surroundings of selected word.

Therefore, we improve the previous *Quantity Cell Graph* by softly connecting surroundings of the quantity-related words to the quantity and named it *Softly Expanded Quantity Cell Graph* G_{soft}. Fig. 3 shows an example. As the distance between the words increases, the weight of the connected edge gradually decreases. This will result in adding a gradually decreasing weight on a vertical and horizontal line of place in adjacency matrix where the connected quantity and the quantity-related words meet. When the window size of related words is k, quantity $N_p = \{n_1, n_2, \cdots, n_l\}$, and quantity-related words $W_p = \{w_1, w_2, \cdots, w_m\}$, the values of matrix $W^{s,t}_{soft}$ containing expanded weight from arbitrary n_s and w_t are calculated using the following equation:

$$W^{s,t}_{soft}[i,j] = \begin{cases} 1 - (\max(|j - w_t|, k) \,/\, k)^2 & \text{if } i = n_s \\ 1 - (\max(|i - n_s|, k) \,/\, k)^2 & \text{if } j = w_t \end{cases} \quad (1)$$

Thus, when $k = 4$, weight will be added to $W^{s,t}_{soft}$ on vertical range of $(w_t - 4, w_t + 4)$ and horizontal range of $(n_s - 4, n_s + 4)$ depending on the distance from n_s and w_t, and those weights will be aggregated on the identical index (i, j), clipped when the aggregated value goes over 1. Therefore, the value of the position (i, j) in adjacency matrix A_{soft} of *Softly Expanded Quantity Cell Graph* is defined as:

$$A_{soft}[i,j] = \min\left(\sum_s \sum_t W^{s,t}_{soft}[i,j], 1\right) \quad (2)$$

For example, in original matrix of Fig. 3, there are no relations between “Amy” and other words. However, in our proposed matrix, the weight propagated from other relations provide softened weights computed by above equation to the row of “Amy”.

C. Question Embedding

To solve MWP, the model needs to find out the information what question requires from the given text. Even when the body text is identical, the solution depends largely on what question is asking for. Thus, in our model, we propose a question embedding z_q added to the input of tree-based decoder directly. With this, decoder can generate an equation based on question embedding without any loss of question information.

To embed the question, we first use BiLSTM to convert text tokens into a node representation. We did not remove the question sentence from the problem sentence, but we used the last hidden state of the BiLSTM after feeding the question sentence as an additional input. Then, that last hidden state z_q is simply concatenated to the graph representation z_g, passing feed forward network to combine those. By inputting question embedding into the global context directly, the model can generate and predict the tree nodes conditioned on question information, estimating $P(E_p|z_g, z_q)$.

IV. EXPERIMENTS

To compare our proposed methods with the original Graph2Tree model, we conduct several experiments and analyzed the results to investigate the effects of each of the components that we applied to our model.

A. Implementation Detail

We trained all the models using the hyperparameters used in the experiments in [19] for the ease of comparison. We used the RoBERTa word embedding with a dimension of 768. The dimensions of the hidden states for layers are set to 384. The batch size and dropout rate are 8 and 0.5 respectively. We use the Adam optimizer with a learning rate of 1e-5 for RoBERTa, 8e-4 for graph-based encoder and tree-based decoder, and used the weight decay of 1e-5. The models are trained for 50 epochs on NVIDIA 2080 Ti GPU. For preprocessing, we used the NLTK package [20] for word tokenizer, and used spaCy [21] for parsing the dependency tree. When G_{dep} is used, the number of GCN head is set to 6 to give 2 heads per graph, whereas when G_{dep} is not used, the number of GCN head is set to 4.

B. Datasets and Evaluation Metrics

Three datasets are used to compare our methods to the Graph2Tree model: SVAMP [19], MAWPS [17], and ASDiv-A [18]. MAWPS contains 2373 MWPs, and ASDiv-A contains 1218 MWPs of arithmetic problems. SVAMP, the dataset created by making variations on both MAWPS and ASDiv-A, contains 1000 MWPs. All of these datasets are in English. We used SVAMP to check if our methods can reach closer to the ideal model that SVAMP expect by considering each variation type, and used MAWPS and ASDiv-A to check whether or not our method can solve existing MWPs generally well. For SVAMP, models are trained on the combined dataset of MAWPS, ASDiv-A, and little amount of the SVAMP then

TABLE IV
RESULT OF EACH MODEL FOR VARIATION TYPE IN SVAMP

		Graph2Tree	with z_q	with G_{dep}	with G_{soft}
Question Sensitivity	Same Obj, Diff Struct	56.4	56.0	**60.5**	56.6
	Diff Obj, Same Struct	36.1	**47.6**	43.8	41.1
	Diff Obj, Diff Struct	56.9	**58.1**	50.6	49.7
	AVG	49.8	**53.9**	51.6	49.1
Reasoning Ability	Add Rel Info	60.9	61.3	**62.8**	61.5
	Change Info	48.3	50.9	50.1	**53.2**
	Invert Operation	82.8	81.8	83.3	**83.7**
	AVG	64.0	64.7	65.4	**66.1**
Structural Invariance	Change order of Obj	**63.6**	61.8	61.8	60.3
	Change order of Phrases	69.8	73.3	73.7	**76.4**
	Add Irrl Info	53.3	52.8	**55.6**	49.6
	AVG	62.2	62.6	**63.7**	62.1

tested purely on SVAMP data that are not used when training, as it is provided on its benchmark. We used 5-fold cross-validation for evaluation and used the solution accuracy as the evaluation metric.

V. RESULTS

We have conducted experiments over five different types of model to compare our proposed methodologies. First, Graph2Tree is an original model proposed by Zhang et al. [15]. Secondly, Graph2Tree + z_q is the model that question embedding z_q is added along with z_g. Third, Graph2Tree + G_{dep} is one that *Dependency Graph* is constructed along with other two graphs of *Quantity Cell Graphs* and *Quantity Comparison Graph*. Fourth, Graph2Tree + G_{soft} changes the original *Quantity Cell Graphs* to *Softly Expanded Quantity Cell Graph*. Lastly, Graph2Tree$^+$ is the one that combines all the methods we propose: Graph2Tree $+z_q + G_{dep} + G_{soft}$.

A. Overall Results

Table II shows the solution accuracy measured on benchmark datasets of MAWPS, ASDIV-A, and SVAMP. From the result, we can observe that the Graph2Tree$^+$ achieved the highest accuracy in every benchmark against other models including original Graph2Tree. This indicates that our graph can surely solve the MWP well, and outperforms the original Graph2Tree, which records state-of-the-art before. Also, we could see that all models which three methods are each applied achieved better results than Graph2Tree, meaning all three methods are effective for overall performance.

B. Results based on Variation Type

To observe whether or not each of our methods works effectively for the variation type that SVAMP suggested, we measured the solution accuracy for each of the variation type they made. Each large categories contains three smaller types of variations. Table III and Table IV shows measured solution accuracy of each variation categories in SVAMP dataset.

1) Question Sensitivity: For Question Sensitivity, we can observe that Graph2Tree$^+$ got better accuracy score than original Graph2Tree model, with difference of 7.2%. This shows that our proposed model can relate information given by the question more than the original when generating equation. Also, we could see that the Graph2Tree with z_q got the highest average result as expected. This shows that the question embedding is effectively working by letting model to generate equation conditioned on the information from question sentence.

2) Reasoning Ability: For Reasoning Ability, Graph2Tree$^+$ got 3.8% higher solution accuracy score compared with Graph2Tree, meaning that our proposed methods actually help model to capture the reasoning of problem text. When comparing with other methods and baseline, Graph2Tree with G_{soft} scored highest accuracy. We assume this is because that the G_{soft} let model to connect surroundings of quantity-related word and thus can gain more semantic information for reasoning.

3) Structural Invariance: For Structural Invariance, Graph2Tree$^+$ got higher score than Graph2Tree with difference of 0.4%, and Graph2Tree with G_{dep} got the highest average score than other two methods. This indicates that *Dependency Graph* helps model to understand structural information by giving information of how words are depending on each other, which is related to the structural characteristic of the natural language. However, we can observe that the difference between Graph2Tree and the Graph2Tree with G_{dep} is only 1.5% meaning that *Dependency Graph* may not enough for model to fully understand the structural information.

VI. CASE STUDY

To check the difference between Graph2Tree model and Graph2Tree$^+$ more sophisticatedly, we carried out a case study. The results of case studies are shown in Table V. In Case 1, the model is required to relate the "peaches" that are in "green" and "red", not "yellow". Graph2Tree$^+$ extracted this successfully from the question by retrieving numbers of "71" and "8", not "7" which is about "yellow peaches". However, Graph2Tree failed to extract the information and considered "7" as a part of the equation.

In case 2, Graph2Tree and Graph2Tree$^+$ generate different results in the type of operation they used. In this question, the model needs to understand the sequence of "collects", "threw away", and "has" for calculating the number of "bottle caps". Graph2Tree generated a wrong result by using "+" whereas

TABLE V
CASE STUDY WITH RESPECT TO VARIATION IN SVAMP

Question: 7 red peaches 71 yellow peaches and 8 green peaches are in the basket. how many more green peaches than red peaches are in the basket? **Variation Type:** Question Sensitivity	
Graph2Tree: 71 - 7	**Graph2Tree$^+$:** 8 - 7
Question: Danny collects bottle caps. He found 63 bottle caps at the park while he threw away 51 old ones. Now he has 33 bottle caps in his collection. How many bottle caps did Danny have at first? **Variation Type:** Reasoning Ability	
Graph2Tree: 33 + 51 + 63	**Graph2Tree$^+$:** 33 + 51 - 63
Question: Danny collects bottle caps and wrappers. He found 66 wrappers and 39 bottle caps at the park. Now he has 16 bottle caps and 68 wrappers in his collection. How many wrappers did Danny have at first? **Variation Type:** Structural Invariance	
Graph2Tree: 16 - 66	**Graph2Tree$^+$:** 68 - 66

Graph2Tree$^+$ generated the right result of using "-". This indicates that Graph2Tree$^+$ understands how reasoning works in the body text.

In Case 3, irrelevant information is added into the body text: "39 bottle caps". Because question asks the number of "wrappers", the model should ignore any numbers that are not relevant to it. Unlike Graph2Tree, Graph2Tree$^+$ understands this and obtained a correct result.

VII. Related Work

Math Word Problem (MWP) is the task of converting short natural language sentences into the simple mathematical equation and solution. The MWP task includes number word problems [22], logic puzzle problems [23], arithmetic word problems [24], [25], algebra word problems [26]–[28], and geometry word problems [29], and have drawn interests from lots of researchers.

Previous MWP methods can be largely categorized into pattern matching based [30]–[34], statistical machine learning based [35]–[37], and semantic parsing based [28], [38], [39]. They were mainly evaluated on small size of datasets, and tend to show low performance on large datasets [40]. To address this problem, recent works employ deep learning based models including recursive neural networks [16], sequence-to-sequence [1], multi-head attentions [41], tree-structure decoders [3], [4], graph-to-tree [12], [15], and graph transformers [13], [14].

Graph representation learning, which uses semantic embedding learning of Graph Neural Networks (GNN), is a research field that has recently attracted attention [5], [6]. GNN has shown a good performance with various structure such as sequence-to-graph [42], graph-to-sequence [43], tree-to-tree [44], graph-to-tree [45], and graph-to-graph [46]. Furthermore, a graph attention network [7] combined with a self-attention-based transformer that performed well in tasks such as neural machine translation [10] and language modeling [48] generated good results in knowledge graph-to-text tasks. Another transformer architecture is the graph transformer [2], which extends the vanilla multi-head attention mechanism to a relation-enhanced global attention mechanism.

VIII. Conclusion

In this paper, we propose an improved version of Graph-to-Tree model [15], Graph2Tree$^+$. We applied three methods to the original model: constructing additional *Dependency Graph*, improving *Quantity Cell Graph* into *Softly Expanded Quantity Cell Graph*, and introducing question embedding. We conducted experiments on three benchmarks of MAWPS, ASDiv-A, and SVAMP, and obtained higher solution accuracy score than state-of-the-art baseline, Graph2Tree. Also, we showed that our methods can effectively deal with the problems even when the question, the structure, or information of the given MWP sentence is modified. We hope that more methods to consider both the linguistic properties and the mathematical properties, such as structural information of the natural language and commutative law of the mathematical equation.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2019-0-00421, Artificial Intelligence Graduate School Program(Sungkyunkwan University)), and by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2021-0-02136, Research and Development of Artificial Intelligence that understands Korean Math Word Problems based on Deep Learning)

References

[1] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a math word problem to a expression tree," presented at the EMNLP, 2018.
[2] T. Chiang, and Y. Chen, "Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems," presented at the NAACL, 2019.
[3] Q. Liu, W. Guan, S. Li, and D. Kawahara, "Tree-structured Decoding for Solving Math Word Problems," presented at the EMNLP, 2019.
[4] Z. Xie, and S. Sun, "A Goal-Driven Tree-Structured Neural Model for Math Word Problems," presented at the IJCAI, 2019.
[5] Y. Li, R. Zemel, M. Brockschmidt, and D. Tarlow, "Gated graph sequence neural networks," presented at the ICLR, 2016.
[6] T. N. Kipf, and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," presented at the ICLR, 2017.
[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," presented at the ICLR, 2018.
[8] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," presented at the NeurIPS, 2017.
[9] K. Xu, L. Wu, Z. Wang, Y. Feng, and V. Sheinin, "Sql-to-text generation with graph-to-sequence model," presented at the EMNLP, 2018.
[10] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph Convolutional Encoders for Syntax-aware Neural Machine Translation," presented at the EMNLP, 2017
[11] K. Xu, L. Wu, Z. Wang, Y. Feng, M. Witbrock, and V. Sheinin, "Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks," arXiv:1804.00823, 2018
[12] S. Li, L. Wu, S. Feng, F. Wu, F. Wu, and S. Zhong, "Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem," presented at the EMNLP, 2020.
[13] R. Koncel-Kedziorski, D. Bekal, Y. Luan, M. Lapata, and H. Hajishirzi, "Text generation from knowledge graphs with graph transformers," presented at the NAACL, 2019.
[14] D. Cai, and W. Lam, "Graph Transformer for Graph-to-Sequence Learning," presented at the AAAI, 2020.

[15] 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," presented at the ACL, 2020.

[16] Y. Wang, X. Liu, and S. Shi, "Deep Neural Solver for Math Word Problems," presented at the EMNLP, 2017.

[17] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "MAWPS: A Math Word Problem Repository," presented at the ACL, 2016.

[18] S. Y. Miao, C. C. Liang, and K. Y. Su, "A Diverse Corpus for Evaluating and Developing English Math Word Problem Solvers," presented at the ACL, 2020.

[19] A. Patel, S. Bhattamishra, and N. Goyal, "Are NLP Models really able to Solve Simple Math Word Problems?," presented at the NAACL, 2021.

[20] S. Bird, E. Klein, and E. Loper, "Natural language processing with Python: analyzing text with the natural language toolkit," O'Reilly Media, Inc., 2009.

[21] M. Honnibal, and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," to appear 2017.

[22] S. Shi, Y. Wang, C. Y. Lin, X. Liu, and Y. Rui, "Automatically Solving Number Word Problems by Semantic Parsing and Reasoning," presented at the EMNLP, 2015.

[23] A. Mitra, and C. Baral "Learning to automatically solve logic grid puzzles," presented at the EMNLP, 2015.

[24] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, "Learning to Solve Arithmetic Word Problems with Verb Categorization," presented at the EMNLP, 2014.

[25] S. Roy, and D. Roth, "Solving General Arithmetic Word Problems," presented at the EMNLP, 2015.

[26] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, "Learning to Automatically Solve Algebra Word Problems," presented at the ACL, 2014.

[27] L. Zhou, S. Dai, and L. Chen, "Learn to Solve Algebra Word Problems Using Quadratic Programming," presented at the EMNLP, 2015.

[28] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, "Parsing Algebraic Word Problems into Equations," presented at the TACL, 2015.

[29] M. Seo, H. Hajishirzi, A. Farhadi, O. Etzioni, and C. Malcolm, "Solving Geometry Problems: Combining Text and Diagram Interpretation," presented at the EMNLP, 2015.

[30] D. G. Bobrow, "Natural language input for a computer problem solving system," MIT, 1964.

[31] D. J. Briars, and J. H. Larkin, "An integrated model of skill in solving elementary word problems," Cognition and Instruction, 1984.

[32] M. Yuhui, Z. Ying, C. Guangzuo, R. Yun, and H. Ronghuai, "Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems," in Education Technology and Computer Science (ETCS), IEEE, 2010.

[33] C. Liguda, and T. Pfeiffer, "Modeling Math Word Problems with Augmented Semantic Networks," International Conference on Application of Natural Language to Information Systems, Springer, Berlin, Heidelberg, 2012.

[34] S. Roy, T. Vieira, and D. Roth, "Reasoning about quantities in natural language," TACL, 2015.

[35] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, "Learning to automatically solve algebra word problems. Association for Computational Linguistics," presented at the ACL, 2014.

[36] L. Zhou, S. Dai, and L. Chen, "Learn to solve algebra word problems using quadratic programming," presented at the EMNLP, 2015.

[37] S. Roy, and D. Roth, "Mapping to declarative knowledge for word problem solving," presented at the TACL, 2018.

[38] S. Shi, Y. Wang, C. Y. Lin, X. Liu, and Y. Rui, "Automatically solving number word problems by semantic parsing and reasoning," presented at the EMNLP, 2015.

[39] Y. Zou, and W. Lu. "Text2Math: End-toend parsing text into math expressions," presented at the EMNLP, 2019.

[40] D. Huang, S. Shi, C. Y. Lin, J. Yin, and W. Y. Ma, "How well do computers solve math word problems? large-scale dataset construction and evaluation," presented at the ACL, 2016.

[41] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang, "Modeling intra-relation in math word problems with different functional multi-head attentions," presented at the ACL, 2019.

[42] X. Peng, D. Gildea, and G. Satta, "Amr parsing with cache transition systems," presented at the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[43] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," presented at the ACL, 2018.

[44] X. Chen, C. Liu, and D. Song, "Tree-to-tree neural networks for program translation," presented at the NeurIPS, 2018.

[45] P. Yin, G. Neubig, M. Allamanis, M. Brockschmidt, and A. L. Gaunt, "Learning to represent edits," presented at the ICLR, 2019.

[46] X. Guo, L. Wu, and L. Zhao, "Deep graph translation," arXiv:1805.09980, 2018.

[47] L. H. B. Nguyen, V. Pham and D. Dinh, "Integrating AMR to Neural Machine Translation using Graph Attention Networks," 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), 2020, pp. 158-162, doi: 10.1109/NICS51282.2020.9335896.

[48] B. Zheng, H. Wen, Y. Liang, N. Duan, W. Che, D. Jiang, M. Zhou, and T. Liu "Document Modeling with Graph Attention Networks for Multi-grained Machine Reading Comprehension," presented at the ACL, 2020.