RESEARCH ARTICLE OPEN ACCESS

Transformer Model for Math Word Problem Solving via Dependency-based Word Embedding

Xinxin Li*, Jiawen Wang**
*(School of Computer Science and Technology, Shandong University of Technology, Zibo, China
Email: lixxin2@gmail.com)
** (Cangzhou Technical Colledge, Cangzhou, China
Email: wjw4056385@163.com)

----------**********----------

Abstract:

Most existing approaches for math word problems directly take the problem sequence as the input, ignoring the syntactic and semantic information. In this paper, we propose a transformer neural model for math word problems via dependency-based word embedding, which utilizes the relations between objects, predicates, and quantities in the problem. Our model uses an encoder-decoder framework, where the encoder takes the dependency-based word embedding as the input, and the decoder predicts the math equations for the problems. The experiments show that incorporating dependency information to the model improves the representation ability of word embedding, and achieves better performance.

Keywords — **Math word problem, Transformer network, Dependency-based word embedding**

----------**********----------

I. INTRODUCTION

Solving math word problems is one of the hot topics in artificial intelligence. Since the 1960s, the research of math word problem solving has begun, attracting many artificial intelligence experts. The common approach for math word problems is mapping the problem text to logic forms or equations that infer the answer. Table I shows an example of a math word problem, which is a minus problem.

The approaches for solving math word problems are rule-based at early stage [1]–[3]. These methods use predefined rules to match sequences, syntactic or semantic representations of the questions, and convert them into equations to solve the problem. However, these systems require hand-crafted engineering to produce the rules and lack specific datasets to validate them. Along with the construction of the math word problem datasets, statistical machine learning approaches are applied [4]–[7]. Math word problems are mapped to logic forms, mathematical operation sequences, or equations through extracted features and then calculated to obtain the answers. In recent decades, deep learning methods have achieved good performance in computer vision and natural language processing [8], [9]. Encoder-decoder models, especially sequence-to-sequence learning, have gained better performance than traditional machine learning methods [10]. Sequence-to-sequence learning usually maps the sequence of a question to the sequence of an equation. The attention mechanism is used to increase the interaction between long-distance words and mathematical variables [11]. There are two important components in sequence-to-sequence learning: the encoder and the decoder. The encoder represents transforming math problem sequences into real-valued representations. Recent research shows that structured representation of the input in

an encoder-decoder model improves machine translation and question answering [12].

TABLE I
A MATH WORD PROBLEM

Problem	Sara played 12 basketball games this year. Her team won most of their games. They were defeated during 4 games. How many games did they win?
Equation	X = 12 - 4
Answer	8

Solving math word problems is not a classification problem to obtain the answer directly. It needs to infer an equation, arithmetic formula, or an action sequence as an intermediate step, usually using the syntactic and semantic information in the problem. According to previous research, incorporating graph neural networks and intermediate semantic representation into the model improves final performance [13]–[15]. This paper proposed a Transformer-based neural model for math word problems. We train a dependency-based word embedding model, utilizing the dependency path and relative dependency position between words in the sentence. We conducted experiments on the MAWPS and Math23K datasets. The experimental results show the model with dependency-based word embedding performs better than the sequence-to-sequence models.

II. RELATED WORK

A. Math Word Problems Solving

Rule-based systems are used for solving math word problems at an early stage. The STUDENT system converted the question into a logical expression through predefined rules [1]. The DEDUCOM system extended logical deduction by adding a depth-first search strategy and a recursive algorithm [16]. The CARPS system introduced structured information into the representation of the problem, which can be used to solve the calculus rate problem [2]. In the 1980s, schemas were introduced to represent math problems, containing semantic information about the question types [17]. Riley et al. introduced three kinds of knowledge: problem schema for understanding the semantic knowledge of the problem, action schema for representing the action knowledge of problem-solving, and strategy knowledge for guiding the actions [18]. However, these rule-based methods require hand-crafted rules.

The statistical machine learning methods take the features of a problem sequence or syntax as the input. Kushman used the Log-Likelihood method to find the equation template of the problem and then extracted information from the problem text to fill the numerical slot and quantity slot of the template [4]. Roy proposed a method based on the arithmetic tree to solve math word problems. It obtained the best arithmetic tree through a classification algorithm [19], introduced unit dependency graphs to represent the dependency between quantities [20], and presented declarative rules to transform mathematical concepts such as dimension analysis and subset relations into expressions [21].

Deep learning methods have made great improvements in math word problems. Wang presented a seq2seq approach to map the problems into equation templates [22]. Huang incorporated the copy mechanism that directly copied the quantities in the problems into the equations and the alignment mechanism that aligned the quantities in the problems with the quantities in the equations [23]. Wang compared the performance of BiLSTM, ConvS2S, and Transformer, and employed an ensemble method that selected the highest probability product of these models as the output [24]. Li proposed a variant of the multi-head attention, group-attention method, to fuse different types of information, including global information, the relations between quantities and their contexts, the relations among quantities, and the relations between quantities and questions [25]. These methods take the sequences of math problems as the input, and use attention networks to empower long-distance constraints, but do not apply any constraints to the output equation sequence.

Xie proposed a tree-based encoder-decoder model to generate the arithmetic tree for math problems [26]. In this model, the decoder used a top-down search method. It first predicted the top node, and then the nodes in the tree, whether they are operators or operands, with a depth-first strategy. Chiang mapped math problem solving to operations on a stack to generate the equation [27]. However, these methods don't consider the

syntactic and semantic information of math questions.

B. Word Embedding Approaches

Word embedding refers to the mapping of a word or phrase to a vector. The skipgram method trained the model using the context of the target word instead of only the previous words [28]. To simplify the model complexity, the output layer used Hierarchical Softmax and Negative Sampling to construct the Huffman tree. ELMO trained the word embedding representation according to the semantics of the context words, so that it could better express the specific meaning in this context [29]. GPT uses Transformer network instead of LSTM as a language model, which can better capture long-distance language structures. Then it finetunes the pre-trained language model for specific tasks [30]. BERT trained a bidirectional language model using the Transformer encoder, which achieved good results for various NLP tasks [31].

In the traditional skip-gram method, the context of a word is its neighboring words in the text. Levy proposed a dependency-based word embedding using word modifiers and heads as the training context [32]. Komninos compared three different word embeddings: the skip-gram model, a variant of the skip-gram model that uses the dependency context, and a variant of the skip-gram model that uses additional information from the dependency graph. The experimental results for question type classification, binary sentiment classification, and semantic relation classification revealed that methods based on dependency context embeddings outperformed the baseline model [33]. Li studied Skip-gram, CBOW, and Glove using dependency relations, and conducted experiments on part-of-speech tagging, named entity recognition, and text classification [34]. MacAvaney studied two dependency embeddings: Stanford embedding and Universal embedding, which uses five levels of label types, from unlabeled to Enhanced++. The results show that the dependent word embeddings are useful for word similarity tasks, but not for name entity recognition [35].

III. OUR MODEL

This paper proposes a transformed-based neural model for math word problems. The model is illustrated in Fig. 1. The problem text is mapped into a real vector sequence using a dependency-based word embedding. Then the Transformer encoder-decoder model generate the mathematical equation of the problem. The generated equation is used to calculate the final answer.

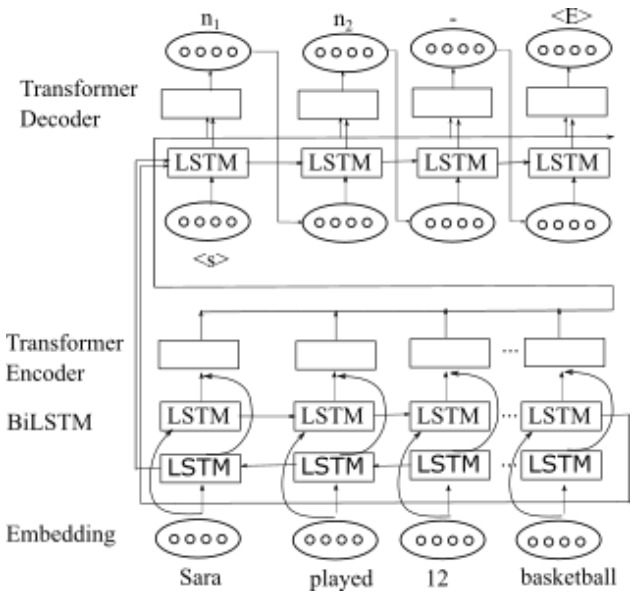


Fig. 1 The architecture of our model

C. Dependency-based Word Embedding

Pre-trained word embeddings have played an important role in improving the performance of neural models. To increase the interaction between entities and quantities in the input sequence, we use dependency-based word embedding as the first layer of our model. Previous work shows that positional information is also useful for many natural language processing tasks [36]. We adopt a positional dependency-based word embedding proposed by Yin, which predicts the target word from neighboring context and dependency-based context. The dependency-based context $\mathbf{c}$ is composed of the dependency context vector $\mathbf{c}_{dep}$ and the positional context vector $\mathbf{c}_{pos}$.

$$\mathbf{c} = \alpha \cdot \mathbf{c}_{pos} + (1-\alpha) \cdot \mathbf{c}_{dep}$$

where α is the weight to balance between dependency context and positional context.

The dependency context vector $\mathbf{c}_{dep}$ represents the information on the dependency path between the target word and its dependency context. Fig. 2 shows an example of a dependency tree. The dependency path for the pair (effect, market) in the figure is $\rightarrow on / IN \rightarrow$. A dependency path $(g_1, g_2, \ldots, g_l)$ is first chosen from the embedding matrix $M_{dep} \in \mathrm{R}^{n \times d}$, and use RNN model the generates the dependency context vector $\mathbf{c}_{dep}$. The positional context vector $\mathbf{c}_{pos}$ are chosen from the embedding matrix $M_{pos} \in \mathrm{R}^{(s-1) \times d}$, where s is the window size.

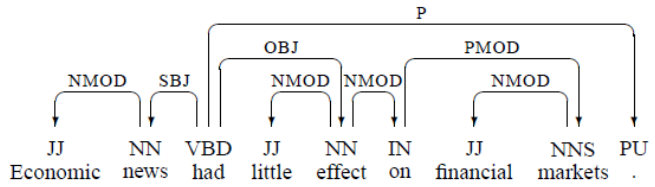


Fig. 2 An example of a dependency tree

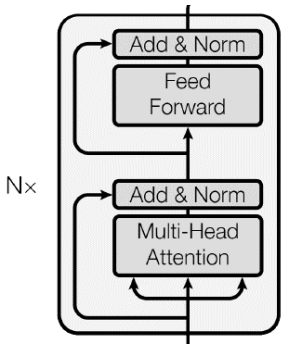
It uses a margin-based method to train the embedding. The loss function is

$$L = \sum \sum max\{S(w_t, c, w_c) - S(w_t', c, w_c) + \phi, 0\}$$

where w_t is the target word, c is the dependency-based context, and w_c is the neighboring context. ϕ is the margin value, $S = (w_c \circ c) \dot{w}_t^T$ is the score function. The loss function is to maximize the difference between golden $S(w_t, c, w_c)$ than random $S(w_t', c, w_c)$.

D. Transformer Layer

The encoder of the transformer contains n identical layers, and each layer contains two sublayers, as shown in Figure 3. Each sublayer is connected using a residual network and then normalized. The output of each sublayer is $Layernorm(x + sublayer(x))$.



Fi. 3 Transformer encoder

The first sublayer is a multi-head self-attention mechanism. It enables the model to learn information from different representation subspaces from these self-attention networks, each of which is scaled dot-product attention.

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O$$

$$where\, head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

The second sublayer is a position-wise fully connected feedforward network.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

IV. EXPERIMENTS

E. Dataset Settings

We perform experiments on two datasets: Math23K and MAWPS [22], [37]. Math23K is a dataset with Chinese math word problems, which contains 23161 problems [22]. MAWPS offers a large collection of data from other existing datasets, e.g., AddSub, SingleOp, MultiArith, SingleEq [4], [5], [19], [38]. It contains 2373 math word problems with one unknown variable. The statistics of the datasets are shown in table II.

TABLE III
STATISTICS OF THE DATASETS

Dataset	MAWPS	Math23K
#questions	2373	23162
#setences	6.3K	70.1K
# single operators	1311	3131
#multiple operators	1062	20031

F. Experimental Results

Neural network models usually need a large amount of training data to achieve good

performance. In this section, we perform experiments to check the performance of our model on different sizes of training data. The experimental results on Math23K dataset are shown in Fig. 3. From the figure, we can see that the accuracy of our model increases with the size of the training data. The model performs better on a large dataset than on a small dataset. It reaches the best performance with all the training data.

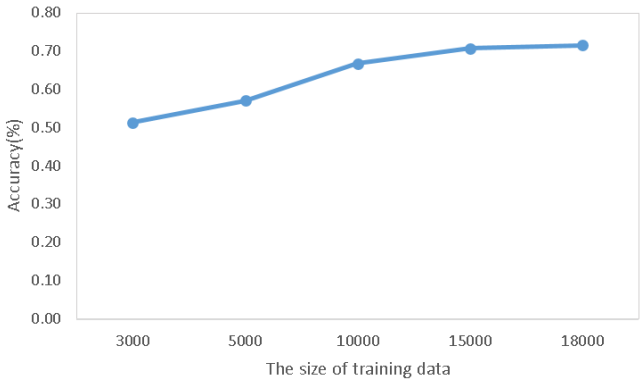


Fig. 3 The accuracy of our model on Math23K with different size of training data

Finally, we evaluate our model with other methods on the MAWPS and Math23K datasets. The results are shown in Table III. It can be seen that our model performs better than previous sequence-to-sequence models. The Graph2Tree model achieves higher accuracy than our model. It uses a graph-based encoder and a tree-based decoder, incorporating quantity cell graph and quantity comparison graph to enhance the relations and order information among quantities.

TABLE IIIII
COMPARISON OF OUR MODEL WITH OTHER METHODS

Models	MAWPS	Math23K
Neural Solver[22]	-	64.7
T-RNN[39]	66.8	66.9
Multi-Head Attention[25]	76.1	69.5
GTS [26]	-	74.3
Graph2Tree [13]	83.7	77.4
Our Model	74.5	71.2

V. CONCLUSION

In this paper, we propose a transformer neural model for math word problems. The transformer layer is used to increase the long-distance relations of words in the input sequence. Our model takes a dependency-based word embedding as its input to represent the relations between entities and quantities. The experimental results show that our model achieves better performance than sequence-to-sequence models. In the future, we will incorporate more structural information to the model.

REFERENCES

[1] D. G. Bobrow, “Natural Language Input for a Computer Problem Solving System,” Department of Mathematics, MIT, Cambridge, 1964. Accessed: Apr. 16, 2019.

[2] E. Charniak, “CARPS: A Program which Solves Calculus Word Problems,” MIT, Cambridge, 1968. Accessed: Apr. 16, 2019.

[3] C. R. Fletcher, “Understanding and Solving Arithmetic Word Problems: A Computer Simulation,” Behavior Research Methods, Instruments, & Computers, vol. 17, no. 5, pp. 565–571, Sep. 1985

[4] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, “Learning to Automatically Solve Algebra Word Problems,” in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, Maryland, Jun. 2014, pp. 271–281

[5] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, “Learning to Solve Arithmetic Word Problems with Verb Categorization,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, Oct. 2014, pp. 523–533

[6] A. Mitra and C. Baral, “Learning To Use Formulas To Solve Simple Arithmetic Problems,” in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, Aug. 2016, pp. 2144–2153

[7] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, “Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems,” in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (volume 1: Long Papers), Vancouver, Canada, Jul. 2017, pp. 158–167

[8] M. Alikhani, P. Sharma, S. Li, R. Soricut, and M. Stone, “Cross-modal coherence modeling for caption generation,” in Proceedings of the 58th annual meeting of the association for computational linguistics, Online, Jul. 2020, pp. 6525–6535

[9] L. Dong, F. Wei, M. Zhou, and K. Xu, “Question Answering over Freebase with Multi-Column Convolutional Neural Networks,” in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, Jul. 2015, pp. 260–269

[10] F. Liu, X. Ren, Y. Liu, H. Wang, and X. Sun, “simNet: Stepwise image-topic merging network for generating detailed and comprehensive image captions,” in Proceedings of the 2018 conference on empirical methods in natural language processing, Brussels, Belgium, Oct. 2018, pp. 137–149

[11] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in 3rd international conference on learning representations, ICLR 2015, san diego, CA, USA, may 7-9, 2015, conference track proceedings, 2015

[12] L. Miculicich, D. Ram, N. Pappas, and J. Henderson, “Document-Level Neural Machine Translation with Hierarchical Attention Networks,” in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, Oct. 2018, pp. 2947–2954

[13] J. Zhang et al., “Graph-to-tree learning for solving math word problems,” in Proceedings of the 58th annual meeting of the association for computational linguistics, Online, Jul. 2020, pp. 3928–3937

[14] D. Huang, J.-G. Yao, C.-Y. Lin, Q. Zhou, and J. Yin, “Using Intermediate Representations to Solve Math Word Problems,” in Proceedings of the 56th Annual Meeting of the Association for

Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia, Jul. 2018, pp. 419–428

[15] C.-C. Liang, S.-H. Tsai, T.-Y. Chang, Y.-C. Lin, and K.-Y. Su, "A Meaning-Based English Math Word Problem Solver with Understanding, Reasoning and Explanation," in Proceedings of COLING 2016, the 26th international conference on computational linguistics: system demonstrations, Osaka, Japan, Dec. 2016, pp. 151–155.

[16] R. F. Simmons, "Natural Language Question-answering Systems: 1969," Commun. ACM, vol. 13, no. 1, pp. 15–30, Jan. 1970

[17] D. J. Briars and J. H. Larkin, "An Integrated Model of Skill in Solving Elementary Word Problems," Cognition and Instruction, vol. 1, no. 3

[18] M. S. Riley, J. G. Greeno, and J. I. Heller, "Development of Children's Problem-Solving Ability in Arithmetic," in The Development of Mathematical Thinking, Orlando, FL: Academic Press, Inc., 1984. Accessed: Apr. 16, 2019

[19] S. Roy and D. Roth, "Solving General Arithmetic Word Problems," in Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, Sep. 2015, pp. 1743–1752

[20] S. Roy and D. Roth, "Unit Dependency Graph and Its Application to Arithmetic Word Problem Solving," in Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, California, USA, 2017, pp. 3082–3088. Accessed: May 04, 2019

[21] S. Roy and D. Roth, "Mapping to Declarative Knowledge for Word Problem Solving," Transactions of the Association for Computational Linguistics, vol. 6, pp. 159–172, Jul. 2018

[22] Y. Wang, X. Liu, and S. Shi, "Deep Neural Solver for Math Word Problems," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, Sep. 2017

[23] D. Huang, J. Liu, C.-Y. Lin, and J. Yin, "Neural Math Word Problem Solver with Reinforcement Learning," in Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, New Mexico, USA, Aug. 2018, pp. 213–223.

[24] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, "Translating a Math Word Problem to a Expression Tree," in Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, Oct. 2018, pp. 1064–1069

[25] 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," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, Jul. 2019, pp. 6162–6167

[26] Z. Xie and S. Sun, "A Goal-Driven Tree-Structured Neural Model for Math Word Problems," in Proceedings of the twenty-eighth international joint conference on artificial intelligence, IJCAI-19, Jul. 2019, pp. 5299–5305

[27] T.-R. Chiang and Y.-N. Chen, "Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems," in Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, Jun. 2019, pp. 2656–2668

[28] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in NIPS, Harrahs and Harveys, Lake Tahoe, Nevada, United States, 2013, pp. 1–9.

[29] M. E. Peters et al., "Deep contextualized word representations," in Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2018, new orleans, louisiana, USA, Jun. 2018, vol. 1, pp. 2227–2237

[30] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers), Minneapolis, Minnesota, 2019, pp. 4171–4186

[32] O. Levy and Y. Goldberg, "Dependency-Based Word Embeddings," in Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, Maryland, Jun. 2014, pp. 302–308

[33] A. Komninos and S. Manandhar, "Dependency based embeddings for sentence classification tasks," in Proceedings of the 2016 conference of the north American chapter of the association for computational linguistics: Human language technologies, San Diego, California, Jun. 2016, pp. 1490–1500

[34] B. Li et al., "Investigating Different Syntactic Context Types and Context Representations for Learning Word Embeddings," in Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, Sep. 2017, pp. 2421–2431

[35] S. MacAvaney and A. Zeldes, "A deeper look into dependency-based word embeddings," in Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Student research workshop, New Orleans, Louisiana, USA, Jun. 2018, pp. 40–45

[36] A. Vaswani et al., "Attention is all you need," in Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, long beach, CA, USA, Dec. 2017, pp. 5998–6008

[37] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi, "Mawps: A Math Word Problem Repository," in Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, Jun. 2016, pp. 1152–1157

[38] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, "Parsing Algebraic Word Problems into Equations," Transactions of the Association for Computational Linguistics, vol. 3, pp. 585–597, Dec. 2015

[39] L. Wang et al., "Template-Based Math Word Problem Solvers with Recursive Neural Networks," in The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, Feb. 2019, pp. 7144–7151