GraphMR: Graph Neural Network for Mathematical Reasoning

Weijie Feng†, Binbin Liu†, Dongpeng Xu*, Qilong Zheng†, Yun Xu†
†University of Science and Technology of China
*University of New Hampshire
{fengwj, robbertl}@mail.ustc.edu.cn
dongpeng.xu@unh.edu
{qlzheng, xuyun}@ustc.edu.cn

Abstract

Mathematical reasoning aims to infer satisfiable solutions based on the given mathematics questions. Previous natural language processing researches have proven the effectiveness of sequence-to-sequence (Seq2Seq) or related variants on mathematics solving. However, few works have been able to explore structural or syntactic information hidden in expressions (e.g., precedence and associativity). This dissertation set out to investigate the usefulness of such untapped information for neural architectures. Firstly, mathematical questions are represented in the format of graphs within syntax analysis. The structured nature of graphs allows them to represent relations of variables or operators while preserving the semantics of the expressions. Having transformed to the new representations, we proposed a graph-to-sequence neural network GraphMR, which can effectively learn the hierarchical information of graphs inputs to solve mathematics and speculate answers. A complete experimental scenario with four classes of mathematical tasks and three Seq2Seq baselines is built to conduct a comprehensive analysis, and results show that GraphMR outperforms others in hidden information learning and mathematics resolving.

1 Introduction

Mathematical reasoning aims to infer satisfactory solutions for the given mathematical problem based on logical rules. It plays a pivotal role in computer algebra (Risch, 1970; Bronstein, 2005; Geddes et al., 1992) (e.g., automated theorem proving), design formulas in numerical programs, and complete scientific computations. With the creation of massive data of logical expressions, neural networks (NNs) based on statistical learning (Rumelhart and McClelland, 1986) have shown success on mathematical reasoning tasks countered to rule-based approaches. For example, Evans et al. (2018); Allamanis et al. (2017) applied tree neural networks to process logic entailment and boolean arithmetic problems. (Lample and Charton, 2019; Piotrowski et al., 2019) released elaborated NNs for symbolic integration and differential equations, shown greater competitiveness than algebra systems.

However, much of the sequence-to-sequence (Seq2Seq) methods up to now (Kushman et al., 2014; Ling et al., 2017; Piotrowski et al., 2019; Saxton et al., 2019) have not explored the structural information behind expressions, such as precedence and associativity. For example, modern Seq2Seq models treat structured symbolic expressions with specific syntax as serialized natural language sentences, which result treat $x+y$ and $y+x$ as different expressions. Moreover, most of these works focus on single tasks (Kaiser and Sutskever, 2016; Trask et al., 2018), lack the capability to handle complex computation.

This study seeks to excavate the implicit information hidden in mathematical problems and examine the capacities of graph-to-sequence architecture to solve large-scope mathematical problems, such as algebra, arithmetic, and polynomial. The first step in this study was to investigate efficient graph substitutes of mathematical expressions. We discuss the representation ability of Abstract Syntax Tree (AST) and Directed Acyclic Graph (DAG) (Thulasiraman and Swamy, 1992), which reveal the structural information of expressions while preserving the semantics of operators and arithmetic entities. Then numeral decomposition strategy is presented to replace graph node normalization of AST/DAG. Subsequently, we introduce GraphMR, a graph-to-sequence approach, which can effectively learn preserved information in graph inputs. To display the performance and generalization ability of our proposal, we conduct complete experiments on four classes of tasks: POLY1, POLY6, MBASIM, and MATHEMATICS. The large-scale evaluations show that GraphMR achieves a better

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3395–3404
November 7–11, 2021. ©2021 Association for Computational Linguistics

performance than state-of-the-art Seq2Seq models. Furthermore, the experiment result shows that DAG achieves the same accuracy on benchmarks with a smaller graph size than AST.

To summarize, main contributions of this dissertation are as follows:

- We discussed two effective graph representations of mathematical problems, i.e., AST and DAG, and proposed a strategy to reduce node feature complexity.
- We introduced GraphMR, a Graph2Seq-based model, to do symbolical reasoning. To the best of our knowledge, this is the first work to analyze mathematical expression with Graph2Seq.
- Detailed and comprehensive experiments on various mathematics tasks were performed. The results showed that GraphMR outperforms state-of-the-art Seq2Seq models.[1]

2 Background

2.1 Mathematical Reasoning with Neural Networks

In recent years, there has been an increasing amount of research on mathematical reasoning using sequence-to-sequence neural networks. Allamanis et al. (2017) tried to use parse trees to represent symbolic expressions and solved them by tree neural networks. A major problem with the parse tree is that it limits the representation capability of the model. Their method can only handle small-size expressions and has an exponential explosion of semantic space to be represented. Evans et al. (2018) proposed a framework called PossibleWorldNet and have shown its ability to solve logical entailment problems. The main disadvantage is that they failed to prove the model's generalization ability in other mathematical problems such as polynomials or arithmetics.

More recently, Lample and Charton (2019); Piotrowski et al. (2019) released elaborated synthesized datasets for symbolic integration and differential equations. Their experiments have shown that Seq2Seq models outperform commercial CAS such as Matlab or Mathematica. However, they considered expressions as sequences of independent symbols, which missed structural information hidden in symbolic expressions.

Conversely, other Seq2Seq related works (Kushman et al., 2014; Ling et al., 2017; Saxton et al., 2019) are more focused on natural language understanding. Their tasks are represented as mathematical query, which is more like neural machine translation.

2.2 Graph-to-Sequence Learning

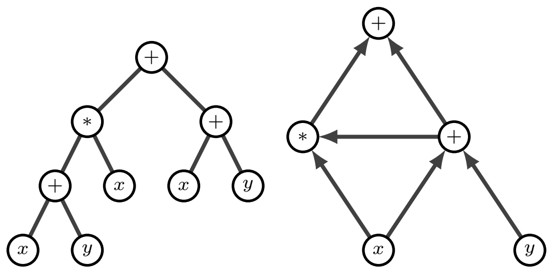
Up to now, a number of studies have demonstrated the effectiveness of graph neural networks (GNNs) (Kipf and Welling, 2017; Gilmer et al., 2017; Hamilton et al., 2017) on non-Euclidean structured data. Bastings et al. (2017) conducted English-German and English-Czech translations with Graph2Seq. They formulate a natural sentence to a dependency tree and encode it with various GNNs to obtain different context vectors. Similarly, Beck et al. (2018) proposed Gated GNN to conduct syntax-based natural machine translation and abstract meaning representations generation.

Recently researchers have applied GNNs to non-graph structured data, such as pictures, videos, and words (Norcliffe-Brown et al., 2018; Liu et al., 2019; Chen et al., 2019). Gao et al. (2019) extended Graph2Seq application scenarios to medical diagnostics to make health stage predictions. Chen et al. (2020) applied the Graph2Seq to natural question generation, which aims to generate a human-readable query based on the given question-answer pair. Shen et al. (2020); Xu et al. (2018) showed the ability of Graph2Seq in video processing, which task aims to generate more grounded and accurate descriptions by linking the generated words with the regions in video frames. However, to the best of our knowledge, there is no systematic research on mathematical reasoning with Graph2Seq.

3 Methodology

To bring out the structural information of mathematics, we firstly introduced two effective graph representations, namely AST and DAG. A numeral decomposition strategy was then discussed to reduce node feature complexity. Following the graph generation process was then briefly described. Finally, GraphMR was depicted to collect and learn the hidden information from input graphs with its encoder and to give out a satisfactory solution through its decoder.

[1] All code and datasets are available at https://github.com/nhpcc502/GraphMR.



(a) Abstract Syntax Tree (b) Directed Acyclic Graph

Figure 1: Different representations for expression $(x + y) * x + (x + y)$.

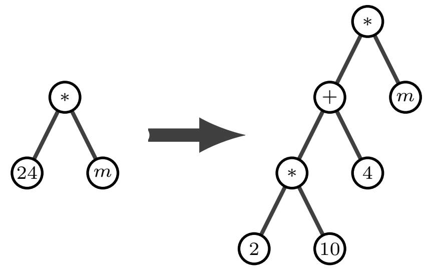


Figure 2: An illustration of numeral decomposition.

3.1 Graph Representation of Mathematics

Traditionally, a mathematical question was represented as a character string, which major advantage is straightforward for humans to read and understand. However, the main problem of the plain approach is that sequence concealed the structural information (e.g., precedence associativity), and Seq2Seq models were unable to utilize them directly. One effective way is to represent expression as Abstract Syntax Tree (AST), in which nodes denote operators, constants, or variables, and edges map the relationship between nodes, as shown in Figure 1(a).

The major problem of representing expressions with AST is that nodes or sub-expressions with the same semantics are repeated in trees, which tends to generate bloated trees when representing long expressions. In order to avoid repetitions, we merged all equal-semantic subtrees or nodes in AST to generate a Directed Acyclic Graph (DAG) version of the mathematical expressions, as shown in Figure 1(b), which merged the subtree $x + y$ and variable x. Similar to AST, operators in DAG with higher precedence order appeared as parents of lower ones. This feature enables AST/DAG to reduce the ambiguity of precedence and associativity, keep the semantics of expressions unchanged, and eliminate assistant characters such as parentheses.

For the purpose of thoroughly compare the performance difference between AST and DAG, we modeled mathematics expressions as these two graphs and performed controlled experiments, respectively.

3.2 Numeral Decomposition

Due to the small total number of characters (alphabet, ten figures, and few operators), Seq2Seq methods are able to encode each character of an expression as a one-hot vector. In contrast, the value domain of constant nodes in graphs is the whole real number, which limits the vectorization of graphs and the normalization of nodes. Furthermore, directly representing different values as different one-hot vectors will weaken the similarity between them. For example, the cosine similarity of the one-hot vector of any two different numerals is zero.

In order to surmount this obstacle, we applied a numeral decomposition strategy during graph generation, which decomposes a big number or decimal into several different parts by adding few bases (e.g., 10, 0.1), as shown in Figure 2. This strategy has two main advantages: i) Effectively narrowed the value domain of constant nodes to $[0, 9]$. ii) Made correlations between different figures and allowed NNs to learn the relationship. Although the application of numerical decomposition led to an increase in the size of the graph, subsequent experiments showed that its effectiveness in both model learning and accuracy improvement.

3.3 Graph Generation Process

Two types of mathematical questions were used in experiments: i) Purely symbolic expressions were our primary concern for their easy conversion to graphs. ii) More common mathematics described by natural language was also examined, and we designed an extractor with numerous regular rules to extract symbolic expressions from questions first. Considering the representation ability of AST/DAG, few specific types of questions such as series, comparison, and sorting were omitted in experiments.

Once symbolic expressions were obtained, there are a number of tools available for transforming them into AST (e.g., AST module[2]). To generate the DAG, we rewritten the AST module to traverse the AST and compared its subtrees or leaves to

[2] https://docs.python.org/3/library/ast.html

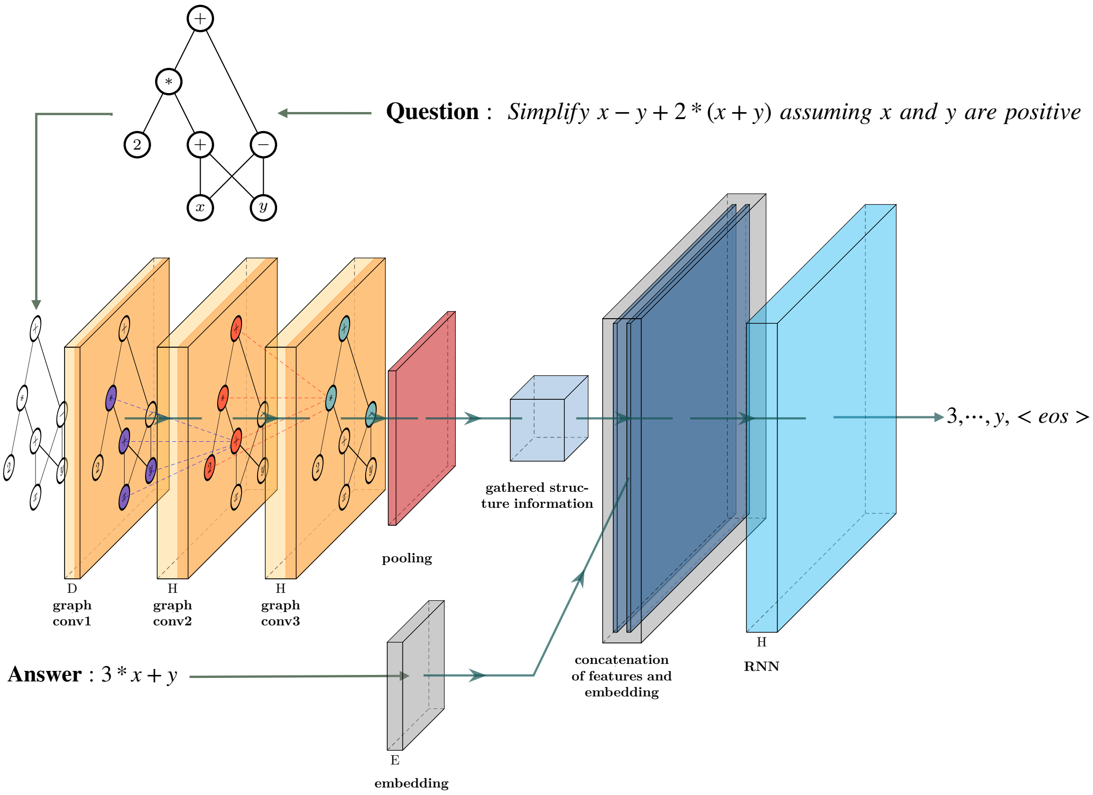

+
*
2
+
−
x
y
Question : Simplify x − y + 2 * (x + y) assuming x and y are positive
3,···, y, < eos >
gathered structure information
pooling
D
graph conv1
H
graph conv2
H
graph conv3
Answer : 3 * x + y
E
embedding
concatenation of features and embedding
H
RNN

Figure 3: Framework of GraphMR. It includes an encoder and a decoder. The encoder is composited by multiple graph convolutional layers and a global pooling layer. The decoder is composited by an embedding layer and an RNN layer.

perform a merger. It is worth noticing that there are usually special operators in expressions that were regarded as both unary and binary operators, and we counted them as different operators and assign corresponding node features according to their in-degree in AST/DAG.

3.4 Architecture of GraphMR

Based on the graph representation of expression, GraphMR was developed for symbolic reasoning based on Graph2Seq. The primary insight of the proposal resides in the information gathering strategies in existing deep learning models. A general overview of the model is shown in Figure 3. Similar to Seq2Seq, GraphMR was composited by a graph encoder and a sequence decoder.

Graph convolutional encoder For a directed graph, an inductive node embedding algorithm was applied to generates bi-directional node embeddings by aggregating information from nodes' local forward and backward neighborhoods (i.e., parents and children) within k hops. More formally, given $G = (V, E)$, its structural information for each node $v \in V$ was collected from two different directions according to the equations below:

$$\mathbf{h}_k^{\vdash} = \mathcal{A}(\{\sigma(\mathbf{W}^{(l)} \cdot \mathbf{X}_{u\vdash}^{k} + \mathbf{b}^{(l)}), u \in \mathcal{N}_{\vdash}(v)\})$$

$$\mathbf{h}_k^{\dashv} = \mathcal{A}(\{\sigma(\mathbf{W}^{(l)} \cdot \mathbf{X}_{u\dashv}^{k} + \mathbf{b}^{(l)}), u \in \mathcal{N}_{\dashv}(v)\})$$

where $\mathcal{N}_{\vdash}(v)$ represents the set of forward neighbors of node v, and $\mathcal{N}_{\dashv}$ represents the set of backward neighbors. $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are randomly initialized learnable parameters for graph convolutional layer l. $\mathbf{X}_{u\vdash}^{k}$ and $\mathbf{X}_{u\dashv}^{k}$ are feature vectors of node u. $\sigma(\cdot)$ represents the non-linear activation function of the model (e.g., ReLU), and $\mathcal{A}(\cdot)$ is an aggregator used for collecting information from neighbors. Figure 4 illustrates the process of node embedding in graph encoder by an example. In the subsequent experiments, k was set to 1 considering the fact that each internal operator is naturally directed correlates to its children and parents only.

Considering neighborhoods of one node have no natural ordering, the aggregation function should be invariant to the ordering of its inputs, ensuring that models can be trained to an arbitrarily ordered set of node-neighborhood features. Besides, the strategy for aggregating information can affect the

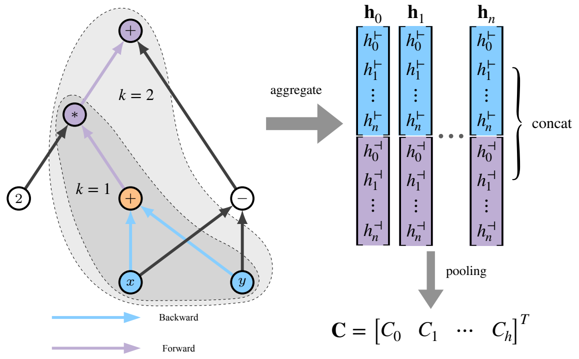


Figure 4: Illustration for node embedding in encoder with different k hops. The encoder collects information from its descendants (blue flow) and ancestors (purple flow), then concatenates those information to form node embedding for the current node (orange one). In graph level, a global pooling operation is applied to generate graph level representation $\mathbf{C}$.

results of one model to some degree. In this work, structural information was aggregated by

$$\mathbf{h}_v = \frac{1}{|\mathcal{N}(v)|} \sum \mathbf{h}_u, u \in \mathcal{N}(v)$$

After aggregating node information, we concatenated them to produce an embedding for current node v as follow:

$$\mathbf{h}_v = \text{concat}(\mathbf{h}_v^{\vdash}, \mathbf{h}_v^{\dashv})$$

To convey high-level graph information to the downstream decoder, aggregating node level embeddings to graph level is essential for a graph task. We used a global sum pooling to obtain this graph level information, which was formulated as:

$$\mathbf{C} = \sum_v \mathbf{h}_v$$

in which $\mathbf{h}_v$ was the embedded node feature matrix generated by the strategy described above. The generated graph-level embedding $\mathbf{C}$ would be sent to the decoder to guide the prediction process.

Sequence decoder Once the convolutional encoder aggregated node information and captured the entire graph to generate graph-level embedding, its output tensor $\mathbf{C}$ would be treated as heuristic information and fed into the sequence decoder. In the decoding stage, an embedding layer was used to embed all previous token sequences $y = y_0 y_1 \cdots y_{t-1}$ and generate an embedding vector $\mathbf{e}$. With the graph embedding $\mathbf{C}$ and sequence embedding $\mathbf{e}^{(t)}$ in time step t, the decoder was able to predict next tokens y_t by:

$$\begin{aligned} \mathbf{h}^{(t)} &= \text{RNN}(\text{concat}(\mathbf{e}^{(t)}, \mathbf{C}), \mathbf{h}^{(t-1)}) \\ y_t &= \text{FC}(\mathbf{e}^{(t)}, \mathbf{h}^{(t)}, \mathbf{C}) \end{aligned}$$

where $\mathbf{h}^{(t)}$ represents hidden state in time step t, and $\text{FC}(\cdot)$ represents the fully connected layer. Notably, we initialize the hidden state with the global graph representation $\mathbf{C}$, i.e., $\mathbf{h}^{(0)} = \mathbf{C}$, as mostly Seq2Seq models do.

4 Experiments

4.1 Experimental Setup

Baselines We compared the following peer Seq2Seq works in experiments: i) GRU-based Seq2Seq model (Cho et al., 2014). As the same as the original work, we reproduced its encoder and decoder with a one-layer GRU unit. ii) Seq2Seq model based on Long Short-Term Memory (Sutskever et al., 2014), we implemented the encoder and decoder with four LSTM layers, as the original paper did. iii) Transformer, a dominant architecture in natural language translation. It is entirely made up of linear layers, attention mechanisms, and normalization. We realized a light version of BERT (Devlin et al., 2018), the most popular variant of Transformer.

Tasks Experimental mathematics was categorized into purely symbolic expressions and non-structured mathematical problems.

Three datasets for the symbolic task were collected. The first two came from POLY (Piotrowski et al., 2019), a symbolic rewriting dataset aiming to represent symbolic expression in another form. We selected different complexity levels tasks in POLY to evaluate all models. The easiest one (POLY1) involves only addition and multiplication operations, whereas the most complicated one (POLY6) contains various mathematical tasks, such as the expansion and the combination of expressions. The third dataset intends to simplify Mixed Boolean-Arithmetic (MBA). MBA expressions involve mixed usage of arithmetic and Boolean operations, making it be NP-hard for human and modern symbolic algebra systems to solve. We used the method contributed by (Feng et al., 2020) to generate the MBASIM dataset.

MATHEMATICS Saxton et al. (2019) is a question-answer dataset from a range of question types at roughly school-level difficulty. It contains

		Length			# Variables			# Operators		
		scale	mean	Std.	scale	mean	Std.	scale	mean	Std.
POLY1	Q	$15.00_{\pm 12.00}$	21.87	4.25	$5.00_{\pm 3.00}$	6.72	1.06	$4.00_{\pm 3.00}$	5.72	1.06
	A	$11.00_{\pm 10.00}$	3.13	2.69	$6.00_{\pm 5.00}$	2.07	1.35	$5.00_{\pm 5.00}$	1.07	1.35
POLY6	Q	$24.00_{\pm 23.00}$	26.45	10.89	$7.00_{\pm 6.00}$	7.86	2.72	$6.00_{\pm 6.00}$	6.86	2.72
	A	$50.00_{\pm 49.00}$	10.51	12.07	$25.50_{\pm 24.50}$	5.76	6.03	$24.50_{\pm 24.50}$	4.76	6.03
MBASIMP	Q	$57.00_{\pm 50.00}$	47.95	14.58	$18.00_{\pm 16.00}$	14.57	4.04	$23.00_{\pm 21.00}$	19.14	5.74
	A	$18.00_{\pm 17.00}$	10.01	5.21	$5.50_{\pm 4.50}$	3.26	1.38	$7.00_{\pm 7.00}$	4.21	2.19
MATHEMATICS	Q	$65.00_{\pm 62.00}$	19.68	16.71	$26.00_{\pm 24.00}$	6.44	6.56	$25.50_{\pm 25.50}$	5.60	6.56
	A	$15.00_{\pm 14.00}$	5.18	3.20	$6.00_{\pm 5.00}$	1.73	1.02	$5.00_{\pm 5.00}$	0.99	1.09

Table 1: Statistic of the experimental datasets. Q indicates input expressions, and A indicates expected results. Three metrics of experimental datasets are the length of expressions, the number of variables (including constants, repeat counted), and the number of operators, respectively. Note that arithmetic-independent words in MATHEMATICS are not counted.

numerous types of non-structured mathematical problems, in which we filtered the comparison and probability part and selected graph-convertible mathematical problems (11 types in total) as our experiments materials, such as linear equation, polynomial roots, surds, and differentiation.

These four datasets coverage most types of mathematical problems and are differentiated in terms of solution difficulty. Ten thousand samples were prepared for each symbolic expression task, and a total of one billion samples were generated for Mathematics. Table 1 illustrates the summary statistics, and Table 2 given examples for each dataset. It can be seen that the number of variables and operators fluctuates considerably, which reflects the difficulty of problem-solving to some extent.

Settings All models were implemented with PyTorch (Paszke et al., 2019) and PyTorch Geometric (Fey and Lenssen, 2019). All baselines were parameterized as their authors did, whereas the hyper-parameters of GraphMR were determined by grid search. Adam (Kingma and Ba, 2014) was imported to optimized all models, with an initial learning rate of $1e-3$, and dynamic strategies (e.g., ReduceLROnPlateau) were applied to adjust the learning rate. All models were trained for 100 epochs with 128 batch sizes and repeated three times to achieve a reliable result. For each prediction, we used Z3 (De Moura and Bjørner, 2008) to determine whether it is semantically equivalent to the standard answer. All experiments were conducted on a 64-bit Linux machine with 16 AMD Ryzen Threadripper 1900X 8-Core CPUs (3.80GHz), 64 GB RAM, and 12GB NVIDIA RTX 2080 GPU.

	Question	Answer
POLY1	$((x+x)*x)*1$	$2*x^2$
	$((0*0)*(1+1))$ $*(1+(1+0))$	0
POLY6	x^2+2xy $+y^2+2yz$ $+z^2+2zx$	$(x+y+z)^2$
	$(x+y+z)^3$	$x^3+y^3+z^3$ $+3x^2y+3xy^2$ $+3x^2z+3xz^2$ $+3y^2z+3yz^2$ $+6xyz$
MBASIM	$4(x \wedge \neg y)-\neg x$ $+2(\neg(x \oplus y))+1$	$3x-2y$
	$2(\neg x \vee \neg y \vee z)$ $-(\neg x \wedge \neg z)$ $+(\neg x \wedge \neg y \vee \neg z)$ $-(\neg x \wedge \neg z)-\neg y$	$x \vee y \vee z$
MATHEMATICS	Put together -9562 and 0.5	-9561.5
	Expand $(24*t/12)$ $*153*sqrt(t**4)$, assuming t is positive.	$-306*t**3$

Table 2: Examples from each dataset.

4.2 Result and Analysis

In this subsection, the results of the experiments are discussed along within different running phases, i.e., static analysis of models size and their inputs before program execution, dynamic analysis of the efficiency and cost during the model training and predict phase, and accuracy statistics of models prediction results.

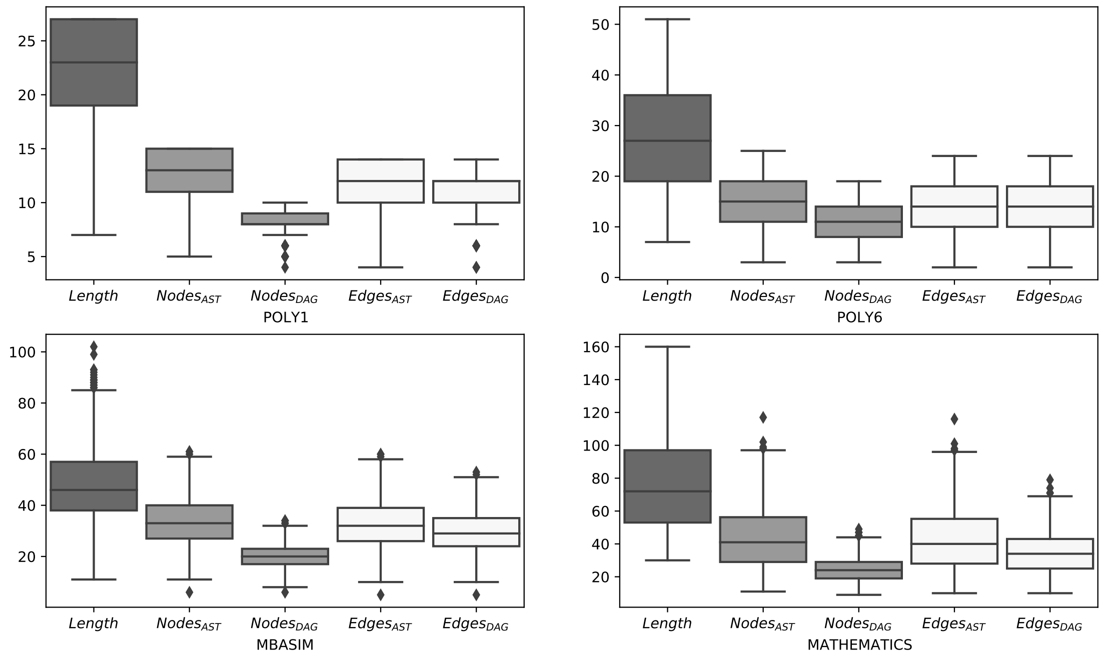


Figure 5: Comparison of the number of nodes and edges when representing four tasks as AST and DAG, where Length denotes the number of characters of raw questions.

	POLY1		POLY6		MBASIM		MATHEMATICS	
	T_{train}	T_{infer}	T_{train}	T_{infer}	T_{train}	T_{infer}	T_{train}	T_{infer}
GRU	**59.7845**	0.0076	217.1197	0.0051	45.3663	0.0076	245.6895	0.0053
LSTM	127.9761	0.0088	316.9887	0.0051	118.8923	0.0253	308.9164	0.0040
Transformer	85.5273	0.0227	169.7207	0.0713	64.2762	0.0522	463.7099	0.0265
GraphMR$_{AST}$	71.2991	0.0057	101.8027	0.0048	**41.1873**	**0.0062**	71.1183	0.0031
GraphMR$_{DAG}$	62.8040	**0.0037**	**93.5846**	**0.0045**	41.5992	0.0063	**57.6448**	**0.0030**

Table 3: Statistic of training time (second per epoch) and inference time (second per sample) for each model on different tasks.

4.2.1 Size of Inputs and Models

Prior to train models, we compared the number of nodes and edges between AST and DAG (with numeral decomposition strategy) and presented the result in Figure 5. It is apparent that DAG can effectively minimize the size of input graphs, which makes it more competitive when the problem complexity in terms of the number of nodes does not allow to use AST anymore. Moreover, compared to baselines, the size of GraphMR's input matrixes (with DAG) are reduced by 52.07%, 48.26%, 32.50%, and 40.55% for POLY1, POLY6, MBASIM, and MATHEMATICS respectively. On the other hand, we piled up multiple GNN layers as the encoder of GraphMR, which allows GraphMR to achieve relative performance with fewer learnable parameters compared to baselines [3].

4.2.2 Models Efficiency and Cost

With smaller inputs and non-recurrent encoder, training time (second per epoch) and inference time (second per sample) of GraphMR are also much less than baselines, as shown in Table 3.

4.2.3 Prediction Results

Accuracy Table 4 shows the evaluation results on accuracy comparing GraphMR against other state-of-the-art baselines. As we can see, graph-based methods outperform other baselines on all symbolic and non-structural tasks. There has no significant difference between baselines and our proposal on the simplest POLY1. AST of this task only contains 15.64 nodes on average, indicates that small-size graphs may lack adequate structural information for exploiting graph-based methods' advantages to the full. In contrast, GraphMR per-

[3] In this experiments, GRU, LSTM, BERT, and GraphMR have 3.18, 7.38, 4.03, and 2.53 million learnable parameters, respectively.

		POLY1	POLY6	MBASIMP	MATHEMATICS
Baselines	GRU	99.02	75.50	47.97	39.90
	LSTM	93.20	78.23	52.55	41.05
	BERT	98.49	81.07	61.72	76.83
Ours	GraphMR$_{AST}$	99.57	83.41	79.70	77.49
	GraphMR$_{DAG}$	**99.61**	**84.05**	**80.40**	**78.54**

Table 4: Comparative accuracy between GraphMR and baselines on various tasks.

forms better on complex tasks, such as MBASIMP, which has 33.72 nodes on average. For non-structured dataset MATHEMATICS, GraphMR has a marked promotion than baselines, which can be attributed to the fact that GraphMR only fed computation-relevant characters.

The accuracy results for different graph representations show that GraphMR with DAG have the same accuracy as AST, which implies that DAG holds similar information entropy and more compact graph size compared to AST. This finding suggests that DAG is a more appropriate representation of mathematics.

The other interesting finding is that BERT has a similar performance to GraphMR on all tasks. With attention mechanisms, a sentence can be deemed as a fully connected graph and words are treated as nodes. This characteristic makes BERT behaves like a fully connected graph neural network.

Impact of numeral decomposition Turning now to the experimental evidence on the effect of numeral decomposition. It can be seen from Figure 6 that the strategy makes a significant boost on accuracy.

A possible explanation for this phenomenon might be that numeral decomposition reduces the diversity of nodes' features, increases each feature's frequency, and allows the model to sample features more smoothly.

5 Conclusion

This essay has studied the symbolic reasoning problem through graph neural networks. AST and DAG were discussed to represent mathematical questions, which can preserve the structural and semantic information of the expressions. Then we introduced a novel method, GraphMR, based on Graph2Seq to address mathematical reasoning problems. Evaluation results show that GraphMR outperforms state-of-the-art Seq2Seq methods in model size and accuracy. Moreover, it can cope with various mathematics tasks. A limitation of this work is that the answers of mathematical problems were treated as character sequences. Further research could also be conducted to model serialized answers as graphs.

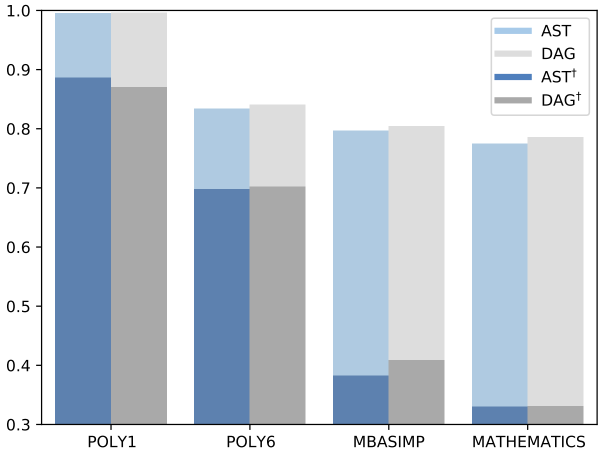


Figure 6: Comparison of accuracies of GraphMR under different configuration. AST† and DAG† represents prediction accuracies without numeral decomposition.

Acknowledgments

We thank team members from AnHui Province Key Laboratory of High Performance Computing (USTC) and UNH SoftSec group for their helpful suggestions.

References

Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. 2017. Learning continuous semantic representations of symbolic expressions. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 80–88. PMLR.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*, pages 1957–1967. Association for Computational Linguistics.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 273–283, Melbourne, Australia. Association for Computational Linguistics.

Manuel Bronstein. 2005. Symbolic integration i-transcendental functions.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2019. Deep iterative and adaptive learning for graph neural networks. *CoRR*, abs/1912.07832.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Reinforcement learning based graph-to-sequence model for natural question generation. In *ICLR*. OpenReview.net.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. Cite arxiv:1406.1078Comment: EMNLP 2014.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. Cite arxiv:1810.04805.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? In *ICLR (Poster)*. OpenReview.net.

Weijie Feng, Binbin Liu, Dongpeng Xu, Qilong Zheng, and Yun Xu. 2020. Neureduce: Reducing mixed boolean-arithmetic expressions by recurrent neural network. In *EMNLP (Findings)*, pages 635–644. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*.

Yuyang Gao, Lingfei Wu, Houman Homayoun, and Liang Zhao. 2019. Dyngraph2seq: Dynamic-graph-to-sequence interpretable learning for health stage prediction in online health forums. In *ICDM*, pages 1042–1047. IEEE.

Keith Geddes, S. Czapor, and George Labahn. 1992. *Algorithms for Computer Algebra*.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org.

William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA. Curran Associates Inc.

Lukasz Kaiser and Ilya Sutskever. 2016. Neural gpus learn algorithms. In *ICLR (Poster)*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Guillaume Lample and François Charton. 2019. Deep learning for symbolic mathematics. *CoRR*, abs/1912.01412.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to generate questions by learning what not to generate. *CoRR*, abs/1902.10418.

Will Norcliffe-Brown, Stathis Vafeias, and Sarah Parisot. 2018. Learning conditioned graph structures for interpretable visual question answering. In *NeurIPS*, pages 8344–8353.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Bartosz Piotrowski, Josef Urban, Chad E. Brown, and Cezary Kaliszyk. 2019. Can neural networks learn symbolic rewriting? *CoRR*, abs/1911.04873.

Robert Risch. 1970. The solution of the problem of integration in finite terms. *Bulletin of The American Mathematical Society - BULL AMER MATH SOC*, 76.

D. E. Rumelhart and J. L. McClelland. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. Cite arxiv:1904.01557.

Kai Shen, Lingfei Wu, Fangli Xu, Siliang Tang, Jun Xiao, and Yueting Zhuang. 2020. Hierarchical attention based spatial-temporal graph-to-sequence learning for grounded video description. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 941–947. International Joint Conferences on Artificial Intelligence Organization. Main track.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

K. Thulasiraman and M. N. S. Swamy. 1992. *Graphs: Theory and Algorithms*. John Wiley & Sons, Inc., USA.

Andrew Trask, Felix Hill, Scott E. Reed, Jack W. Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. In *NeurIPS*, pages 8046–8055.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. Cite arxiv:1804.00823Comment: 16 pages, 3 figures, 4 tables.