Neural Mathematical Solver with Enhanced Formula Structure

Zhenya Huang[1], Qi Liu[1], Weibo Gao[1], Jinze Wu[1], Yu Yin[1], Hao Wang[1], Enhong Chen[1,*]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology,
University of Science and Technology of China,
{huangzhy,weibogao,hxwjz,yxonic,wanghao3}@mail.ustc.edu.cn; {qiliuql,cheneh}@ustc.edu.cn

ABSTRACT

Automatically answering mathematical problems is a challenging task since it requires not only the ability of linguistic understanding but also mathematical comprehension. Existing studies usually explore solutions on the elementary math word problems that aim to understand the questions described in natural language narratives, which are not capable of solving more general problems containing structural formulas. To this end, in this paper, we propose a novel *N*eural *M*athematical *S*olver (NMS) with enhanced formula structures. Specifically, we first frame the formulas in a certain problem as a TEX dependency graph to preserve formula-enriched structures. Then, we design a formula graph network (FGN) to capture its mathematical relations. Next, we develop a novel architecture with two GRU models, connecting tokens from both word space and formula space together, to learn the linguistic semantics for the answers. Extensive experiments on a large-scale dataset demonstrate that NMS not only achieves better answer prediction but also visualizes reasonable mathematical representations of problems.

KEYWORDS

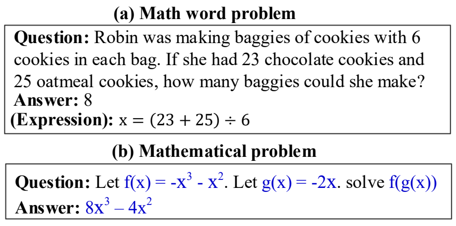
Mathematical problem; Formula; Mathematical comprehension

ACM Reference Format:
Zhenya Huang, Qi Liu, Weibo Gao, Jinze Wu, Yu Yin, Hao Wang, Enhong Chen. 2020. Neural Mathematical Solver with Enhanced Formula Structure. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25-30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401227

1 INTRODUCTION

Letting computers learn and apply knowledge to answer mathematical problems is a crucial issue, which has attracted much attention in information retrieval and natural language processing for a long history [15]. It is challenging as it requires not only the ability of linguistic understanding but also mathematical comprehension [9].

Towards this goal, researchers have explored many possibilities in solving math word problems which is an elementary type of mathematical problems [15]. Figure 1(a) presents an example. It is


(a) Math word problem

Question: Robin was making baggies of cookies with 6 cookies in each bag. If she had 23 chocolate cookies and 25 oatmeal cookies, how many baggies could she make?
Answer: 8
(Expression): x = (23 + 25) ÷ 6

(b) Mathematical problem

Question: Let $f(x) = -x^3 - x^2$. Let $g(x) = -2x$. solve $f(g(x))$
Answer: $8x^3 - 4x^2$

Figure 1: Math word problem vs. Mathematical problem.

described in natural language narratives (Robin was...) and asks a question (how many ...?). To solve this problem, readers translate it in into an expression form for the answer (8). During this process, we are required to identify the necessary numbers (e.g., 23 and 25), and then infer their relation with corresponding operator (e.g., "+") by understanding the question. Along this line, research works have attempted many methods, such as statistical learning [7], and deep learning models [13]. More works can be found in the survey [15].

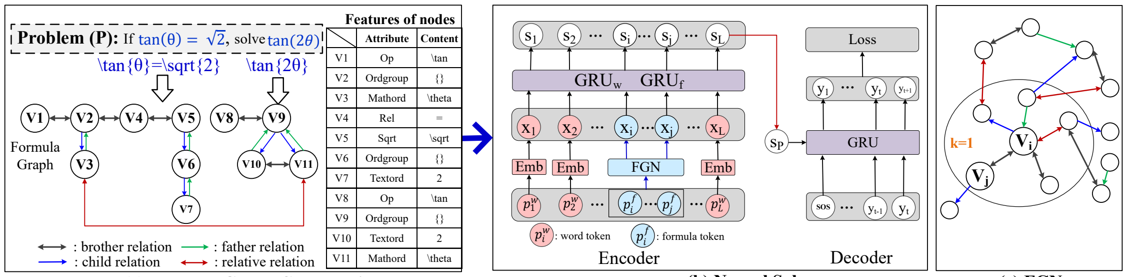
Although great success have been made in this task, it is quite elementary because the solution only requires the linguistic abilities including semantic understanding and operator extraction [15]. Actually, it is far from general mathematical problems (which accord with high school level students), where a typical problem is shown in Figure 1(b). Obviously, it is harder for generating the answer since such problem contains many complicated but informative formulas (marked "blue"), which asks us to make mathematical comprehension. In this paper, we aim to solve this kind of mathematical problems in a principled way, where it is necessary to incorporate the information of formulas for the answers.

However, it is a non-trivial task. First, how to represent such formula-enriched problem remains much open since it is quite difficult to understand the formulas with their free-text format as a token sequence (i.e., word or character) (in Figure 1). If directly applying existing sequential methods [2, 10] for modeling, we only learn their content semantics but ignore much structural information as the prior work [14] suggested. Second, solving the problems because we need to find a unified architecture to learn them with considering both linguistic understanding from their contents of what they ask and mathematical comprehension crossing formulas.

To this end, we propose a novel framework, *N*eural *M*athematical *S*olver (NMS), following the sequence-to sequence architecture (Seq2Seq). Specifically, we first develop an assistant tool to construct formulas in each problem with a TEX dependency graph, which preserves formula-enriched structure. Then, we design a formula graph network (FGN) to capture its mathematical relations. Next, with enhanced formula information, we develop a novel architecture with two GRUs, connecting the problem tokens from both linguistic word space and structural formula space together, to learn the problem semantics for generating answers. We conduct extensive experiments on a large-scale dataset. Experimental results demonstrate that NMS not only achieves better performance of

[*]Corresponding Author.

SIGIR '20, July 25–30, 2020, Virtual Event, China
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8016-4/20/07...$15.00
https://doi.org/10.1145/3397271.3401227



Features of nodes

	Attribute	Content
V1	Op	\tan
V2	Ordgroup	{}
V3	Mathord	\theta
V4	Rel	=
V5	Sqrt	\sqrt
V6	Ordgroup	{}
V7	Textord	2
V8	Op	\tan
V9	Ordgroup	{}
V10	Textord	2
V11	Mathord	\theta

(a) Formula Graph Construction **(b) Neural Solver** **(c) FGN**

Figure 2: NMS framework. (a) Formula graph construction. (b) Model architecture of neural solver. (c) FGN network.

answer prediction but also shows the sophisticated representations for mathematical problems enhanced by formula comprehension. To the best of our knowledge, this is the first attempt to introduce formula learning for solving mathematical problems in the domain.

2 NEURAL MATHEMATICAL SOLVER

2.1 Problem and Overview

Generally, a mathematical problem P is defined as a sequence of L tokens: $P = \{p_1, p_2, \cdots, p_L\}$, where p_i is either a word token (p_i^w) from a vocabulary or a formula token (p_i^f) from the mathematical objects (including quantities or symbols like "+"). There may be multiple formulas in problem P, comprised of some consecutive formula tokens, For example, the problem in Figure 1(b) contains three formulas (e.g., "$f(x) = -x^3 - x^2$"). We denote the formulas in one problem as $F = \{f_1, f_2, \cdots, f_M\}$, where $f_m = \{p_i^f, \cdots, p_j^f\}, 1 \leq i, j \leq L$ denotes a certain formula. Besides, we are also given a answer sequence: $Y = \{y_1, y_2, \cdots, y_T\}$, where token y_j is defined in correspondence with p_i in P. The goal of solving mathematical problems is to learn a model which reads the tokens from P and generate the output $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_T\}$ as accurate as the answer sequence Y.

In this paper, we propose a *N*eural *M*athematical *S*olver (NMS), which is shown in Figure 2. It has two steps: (1) The construction part establishes the formula graph from its free-text format, preserving its formula-enriched structures. (2) The solving part provides a neural solver for generating the answer of problem.

2.2 Formula Graph Construction

As mentioned in Section 1, given a certain problem, it is difficult to learn its formulas included with the free-text format since such treatment overlooks much structural and relational information [14]. Therefore, we should deal with the first challenge of how to present formulas in a structural way, which supports the downstream task of problem solving. Although existing solutions like KaTeX library (https://katex.org/) can produce the formula with an abstract syntax tree (AST), they cannot be directly applied to our formula establishment in mathematical problems due to the following reasons. First, AST contains extra information including original source code and annotations on typesetting and redundant parentheses, which are unnecessary for modeling mathematical structure and dependency, resulting in much larger feature space. Second, AST only treats each formula separately with a tree-based structure, which cannot capture the correlations among multiple formulas in the problem. e.g., the variables "θ" in Figure 2(a) should be connected.

To address these issues, in this paper, we develop an assistant tool to frame the formulas in a problem as a TEX-based dependency graph based on their TEX codes, which as shown in Figure 2(a). It has the following advantages. First, we just treat the mathematical objects as nodes including variables ("θ"), numbers ("2"), and TEX operators ("tan"), which reduces the space compared to AST. Second, we distinguish different relations between nodes by four types, i.e., brother, father, child and relative, to keep the grammar-enriched formula structures. Third, like many attributed networks [11], each node in the formula graph is endowed with some features (i.e., "Attribute", "Content") to enhance their semantics[1]. Therefore, our constructed dependency graph can help represent such formula-enriched mathematical problems.

2.3 Neural Solver

After reconstructing mathematical problems with formula-enriched structures, now we discuss the model details of NMS. Recall Figure 1(b), solving such problems is even harder in that it requires not only the ability of linguistic understanding but also mathematical comprehension. In this paper, we propose a novel neural solver following Seq2Seq architecture with encoder-decoder (Figure 2(b)).

2.3.1 Encoder Architecture. Our solver encoder reads problem sequence and produces it into hidden representation encoding its semantics, which has two modules: (1) an embedding module projects the inputs into hidden space from different token forms; (2) a sentence module learns the problem semantic representations.

Embedding Module. In the embedding module, given a certain problem $P = \{p_1, p_2, \cdots, p_L\}$ consisting of both word tokens (p^w) (red circle) and formula tokens (p^f) (blue circle), we map them separately into an embedded sequence with fixed-length vectors $\{x_1, x_2, \cdots, x_L\}$, denoted as $\text{Emb}_w(\cdot)$ and $\text{FGN}(\cdot)$, respectively.

For the word tokens $\{p_i^w\}$, $\text{Emb}_w(\cdot)$ initializes each of them by an pre-trained word embedding $x_i \in \mathbb{R}^d$ with *word2vec* [6].

For the formula tokens $\{p_i^f\}$, instead of directly modeling with their free-text format, we propose a specialized Formula Graph Network (FGN) (Figure 2(c)) to capture their mathematical relations over the formula graph with comprehension. Specifically, we denote our formula graph as $G = \{V, \mathcal{E}, R\}$. A node $v_i \in V$ represents a mathematical token including variables, numbers or TEX operators etc. $(v_i, v_j, r) \in \mathcal{E}$ is an edge from node v_i to node v_j with a relation $r \in R$ from the set of {brother, father, child, relative}. Moreover, we add a "self" relation into the R to capture nodes' self-loop effects, and therefore, $|R|$=5. Besides, each node is associated with features including its token attribute and content (an example in Figure 2(a)).

Given the formula graph G, FGN introduces the K graph layers to iteratively learn the encoding of every node. At (k+1)-th layer, to

[1]Our tool: https://git.bdaa.pro/math-nlp/latex_rgcn/-/treemaster/Latex

learn the encoding h_i^{k+1} of node v_i, our graph network operation can be specified as a message-passing-receiving mechanism [12]:

$$h_{N_i}^{k+1} = \sum\nolimits_{r \in R} \sum\nolimits_{j \in N_i^r} \frac{1}{|N_i^r|} \mathbf{W}_r^k h_j^k, \tag{1}$$

$$h_i^{k+1} = \sigma(\mathbf{W}^k h_i^k \oplus h_{N_i}^{k+1}), \tag{2}$$

where N_i^r is the subset of N_i consisting of all the neighbors of node v_i with relation r. $\mathbf{W}_r^k$ and $\mathbf{W}^k$ are trainable parameters that transform the node messages. $\oplus$ is vectorial concatenation. $\sigma(\cdot)$ is a non-linear activation function ReLU$(\cdot)$. Generally, FGN has two operations for the encoding. First, Eq. (1) first aggregates the message from all v_i's neighbors. Second, Eq. (2) combines the information of both its neighbors $h_{N_i}^{k+1}$ and node v_i at previous layer h_i^k. It worth mentioning that FGN assumes that neighbors connected by different relations would produce different influences on the node v_i, and thus updates the relation-specific parameters ($\mathbf{W}_r^k$) for learning the embedding of formula token at each layer.

Moreover, at layer 0 of FGN, we initialize each node embedding by assigning its original features from two perspectives: (1) one-hot encoding for "Attribute" embedding; and (2) word2vec embedding for "Content" with its original formula token.

After FGN, we can embed each formula token (p_i^f) based on both its semantics and structural relations, and denote it as its last K-layer's structural embedding: $x_i = h_i^K$.

Sentence Module. Then in the sentence module, we need to learn the linguistic semantics of a problem $\{s_1, s_2, \cdots, s_L\}$ from the embedded sequence $\{x_1, x_2, \cdots, x_L\}$. Note that existing standard methods like LSTM or GRU [2] cannot be directly applied since they just model the sequence in a uniform space. However, our embedded sequence consists of the elements from two space, one from the linguistic word space ($Emb_w(\cdot)$) and the other from the structural formula space ($FGN(\cdot)$). To address this problem, in this paper, we introduce a novel architecture with two GRU models receiving different type inputs but holding one hidden semantic space, connecting with each other. Formally, at l step, our architecture updates the hidden semantic state s_l as:

$$s_l = \begin{cases} \mathrm{GRU}_w(x_l, s_{l-1}; \boldsymbol{\theta}_w) & \text{if} \quad x_l \text{ from word space}, \\ \mathrm{GRU}_f(x_l, s_{l-1}; \boldsymbol{\theta}_f) & \text{if} \quad x_l \text{ from formula space}. \end{cases} \tag{3}$$

At last, the problem is represented as the encoder output: $s_p = s_L$.

2.3.2 Decoder Architecture. The decoder generates the answers with one token y_t at a step t, giving its conditional probability over the problem embedding s_P by the encoder and all the previous outputs $\{y_1, y_2, \cdots, y_{t-1}\}$. In our decoder implementation, we first introduce the standard GRU model [2], i.e., $GRU(\cdot; \boldsymbol{\theta}_D)$ ($\boldsymbol{\theta}_D$ is the trainable parameter) to get the hidden state when decoding, and then generate the answer output with Softmax function.

With our NMS, the overall loss $\mathcal{L}$ on a problem-answer pair can be defined as minimizing the negative log likelihood of the generated token sequences using Adam optimization:

$$\mathcal{L} = \sum\nolimits_{t=1}^{T} -\log P(y_t | y_1, y_2, \cdots, y_{t-1}, s_P). \tag{4}$$

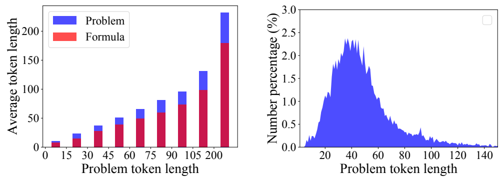
3 EXPERIMENT

3.1 Experimental Dataset and Setup

Dataset. The dataset is collected with mathematical problems for senior high school students from their exercises. Problems are grouped into several knowledge concepts like "Function". We select problems with the most 5 frequent concepts. We conduct data analyses in Table 1 and Figure 3. We observe that the formula tokens take large portions of the whole mathematical problems, i.e., they take nearly 69% of the whole problem tokens on average, and even take larger portions in shorter problems. This demonstrates that formulas have significant effects, which argues that the model should make the comprehension for them to better answer the problem.

Table 1: The statistics of the dataset.

Num. problem	Avg. problem Length	Avg. answer Length	Avg. formula Number	Avg. formula Length
31,500	48.47	8.87	3.46	35.91



(a) Token Length (b) Problem Distribution

Figure 3: Data analyses.

Experimental Setup. In NMS, we set the word embedding size d=100. For FGN, we set the number of graph layers K=2. The dimensions of node embedding in each layer are [200, 150, 100]. We set all GRU models with hidden state size as 50. In the training process, we initialized all the network parameters with a Gaussian distribution (mean 0, standard deviation 0.01). We set the learning rate as 0.001, mini batches as 32 and dropout with 0.1. In the test process, we utilize the beam search with size 3 for the generation.

Baseline and Evaluation. Since there are few attempts which directly solve this mathematical problems, we introduce several representative Seq2Seq models. The first three are *Seq2Seq-GRU*, *Seq2Seq-BiGRU*, *Seq2Seq-RMC*, which have GRU [2], BiGRU [4] and RMC [8] in encoders, respectively. Then, we select *Seq2Seq-Attn* with attention network [1] and the state-of-the-art Transformer [10]. We also introduce a variant of NMS without FGN to highlight the effectiveness of formula structures, denoted as NMS-F. It directly model the formula tokens as their original free-texts.

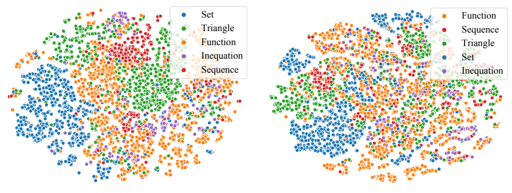
For the evaluation protocols, we partition the data into several training/testing sets with the ratios as 60%/40%, 70%/30%, 80%/20%, 90%/10%, respectively to validate the model performance at different data sparsities. For each training set, we sample 10% data for validation. Then since our task can be treated as a generation task, we select the widely-used metrics including ACC (probability of correct answer) [9], BLEU (average result of BLEU-1, BLEU-2 and BLEU-3), ROUGE (ROUGE-L) for model evaluation.

3.2 Experimental Results

Answer performance. Table 2 reports the results of all models for solving mathematical problems. There are several observations. First, NMS consistently performs the best at all data sparsities, which demonstrates NMS can effectively capture the mathematical relations of formula structure for the task. Second, Transformer and Seq2Seq-BiGRU perform better than other baselines since they all design sophisticated encoders to learn the problem semantics in encoders. Third, RMC performs not very well. This is probably

Table 2: The overall performance of problem solving.

Training/test ratio	60%/40%			70%/30%			80%/20%			90%/10%		
Metric	ACC	BLEU	ROUGE	ACC	BLEU	ROUGE	ACC	BLEU	ROUGE	ACC	BLEU	ROUGE
Seq2Seq-GRU	27.94%	27.78	52.10	28.72%	28.43	52.89	29.38%	29.68	52.99	30.25%	30.50	55.37
Seq2Seq-BiGRU	30.40%	31.78	55.83	30.85%	32.09	56.94	30.47%	31.89	57.05	32.87%	33.85	57.70
Seq2Seq-RMC	26.38%	26.32	49.06	26.50%	27.01	49.81	26.82%	26.98	50.53	27.51%	27.66	51.47
Seq2Seq-Attn	29.59%	30.16	54.95	30.58%	31.48	55.15	30.94%	32.07	56.42	31.13%	31.91	55.59
Transformer	30.71%	32.15	55.81	31.31%	32.90	55.85	32.20%	33.52	56.12	32.32%	34.82	57.38
NMS-F	29.94%	31.24	53.71	30.78%	32.99	55.91	31.86%	33.54	56.96	32.16%	33.62	57.98
NMS	**31.85%**	**33.09**	**55.61**	**32.14%**	**33.88**	**57.22**	**33.65%**	**34.08**	**57.55**	**34.21%**	**36.67**	**59.93**



(a) NMS (b) Seq2Seq-BiGRU

Figure 4: Problem visualization with knowledge concepts.

because introducing memory modules may bring many parameters for the model. Last, we can see NMS-F performs better than GRU, which means it is necessary to treat word and formula tokens in different space separately for the problem solving. Moreover, NMS achieves even better since it capture the formulas' structural relations rather than just treating them as the free-texts.

Problem correlation analysis. As mentioned, by incorporating formula structure into problem embedding, NMS is able to discover the latent correlation among problems, which can support many applications, such as problem recommendation [3]. To demonstrate this task, we randomly sample 5000 mathematical problems, and project their final embeddings, i.e., s_p in Figure 2(b), into 2D space by t-SNE [5]. We also introduce the embedding results learned by Seq2Seq-BiGRU as comparison. We label the problems with their knowledge concepts using different colors. Figure 4 shows the visualizations. First, problems with same concepts learned by NMS are easier to be grouped, meaning that they are closer in the hidden space. This demonstrates that NMS can learn better problem embeddings by making formula comprehension. Moreover, "Set" problems are nearly grouped into one cluster since they have simple formula structures. However, "Function" problems are scattered because they usually incorporate many types of formulas, causing different patterns, e.g., exponential function and logarithmic function have different formula structures, which cannot be grouped.

Discussion. There are some future directions. First, although NMS can capture the formula structural relations for mathematical problems, they sometimes cannot predict quantities effectively. For example, it is difficult to distinguish the numerical difference between $\frac{1}{2}$ (1, 2) and $\frac{11}{22}$ (11, 22) when the problems are not long. For this issue, we may need another reasoning ability to learn problem logic [9]. Second, we will design different graph networks for learning formula structure. Third, we will consider more specific structures (e.g., figure) of more complex problems (e.g., "geometry") for the answers. We hope this work can lead to more studies.

4 CONCLUSION

In this paper, we proposed a novel *N*eural *M*athematical *S*olver (NMS) to answer mathematical problems. Specifically, to preserve formula structure, we first developed an assistant tool to construct $\TeX$ based formula dependency graph to represent each problem. Then we designed a FGN to capture its mathematical relations. Next, we developed a novel model with two GRUs to learn the problem semantics for answers, connecting both word space and formula space. Experimental results demonstrated the effectiveness of NMS.

Acknowledgements. This research was supported by grants from the National Natural Science Foundation of China (Grants No. 61922073, 61672483, U1605251, 61727809).

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015.*

[2] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014.*

[3] Zhenya Huang, Qi Liu, Chengxiang Zhai, Yu Yin, Enhong Chen, Weibo Gao, and Guoping Hu. 2019. Exploring Multi-Objective Exercise Recommendations in Online Education Systems. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management.* 1261–1270.

[4] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[5] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems.* 3111–3119.

[7] Subhro Roy and Dan Roth. 2015. Solving General Arithmetic Word Problems. In *Proceedings of the 2015 Conference on EMNLP.* 1743–1752.

[8] Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. 2018. Relational recurrent neural networks. In *Advances in neural information processing systems.* 7299–7310.

[9] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557* (2019).

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems.* 5998–6008.

[11] Hao Wang, Enhong Chen, Qi Liu, Tong Xu, Dongfang Du, Wen Su, and Xiaopeng Zhang. 2018. A United Approach to Learning Sparse Attributed Network Embedding. In *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 557–566.

[12] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. 2019. MCNE: An End-to-End Framework for Learning Multiple Conditional Network Representations of Social Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1064–1072.

[13] Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence.* AAAI Press, 5299–5305.

[14] Yu Yin, Zhenya Huang, Enhong Chen, Qi Liu, Fuzheng Zhang, Xing Xie, and Guoping Hu. 2018. Transcribing content from structural images with spotlight mechanism. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2643–2652.

[15] Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence* (2019).