Solving Math Word Problems Automatically with Heterogeneous Line Graph Transformer for Online Learning

Zijian Hu
Southeast University
Nanjing, China
zijian@seu.edu.cn

Meng Jiang
University of Notre Dame
Notre Dame, Indiana, USA
mjiang2@nd.edu

ABSTRACT
This paper describes the design and implementation of a new machine learning model for online learning systems. We aim at improving the intelligent level of the systems by enabling an automated math word problem solver which can support a wide range of functions such as homework correction, difficulty estimation, and priority recommendation. We originally planned to employ existing models but realized that they processed a math word problem as a sequence or a homogeneous graph of tokens. Relationships between the multiple types of tokens such as entity, unit, rate, and number were ignored. We decided to design and implement a novel model to use such relational data to bridge the information gap between human-readable language and machine-understandable logical form. We propose a heterogeneous line graph transformer (HLGT) model that constructs a heterogeneous line graph via semantic role labeling on math word problems and then perform node representation learning aware of edge types. We add numerical comparison as an auxiliary task to improve model training for real-world use. Experimental results show that the proposed model achieves a better performance than existing models and suggest that it is still far below human performance. Information utilization and knowledge discovery is continuously needed to improve the online learning systems.

1 INTRODUCTION
Online learning systems have shown rapid growth in recent years, such as coursera.org, KhanAcademy.org and study.163.com. Statistics from Coursera indicate that over 77 million users are currently learning on the platform. With this trend comes an increasing demand for the intelligence level of online education systems [1, 24, 27]. Thousands of learning functions have been developed on these systems to satisfy personalized needs.

Building automated math word problem (MWP) solver provides a significant test for the intellectual abilities of these applications [16]. On one hand, it evaluates whether the problem representations in online learning systems could understand and work with numbers [10, 23, 42]. On the other hand, it brings new possibilities for downstream applications, e.g., answering new problems that were not seen in the database [28]. However, the performance of the current MWP solver is far from providing sufficient support for different question-based applications. That is mainly because the learned representations of educational questions is not informative enough [32, 44]. It is appealing to provide a MWP representation enhancement module to improve the performance of the systems.

As a fundamental issue to test the intelligence of online education systems, automated MWP solver has attracted lots of research interest. At the initial stage, statistical machine learning and semantic parsing methods were mainstream solutions [13, 18, 20, 30, 33], with the objective of mapping the sentences of problem statements into structured logic representations so as to facilitate quantitative reasoning. Recently, with the availability of large-scale training data, people attempted to exploit the expressive power of deep learning models. The deep learning models achieved better performance than previous methods because it avoided manually-crafted features and leverage the advantage of end-to-end mechanism [7, 15, 38, 39].

Because mathematical expressions can be represented as binary trees, researchers have proposed seq2tree architecture that showed superior performance [41]. To enrich the question representations, Graph2Tree architecture was developed to build direct connections between numerals and entities in questions [32, 44]. However, existing models mainly focus on the inference ability while the problem is often simply encoded as a sequence of words by one single vector. There is insufficient information available for decoding because the logical information hidden behind the structure among words is usually ignored. Take the structural relations between “ratio”, “transported goods”, “rest”, “is” and “3:7” in Figure 1 as example. According to the rule of semantic role labeling (SRL), this clause has “is”, “ratio”, and “3:7” as its root, agent, and patient nodes, respectively [35]. In addition, “transported goods” and “rest” stand for the extent. Based on the semantic role labels, the model is expected to learn that “3:7” is a “ratio”. Then, the model would be able to realize that the “ratio” describes the relationship between “transported goods” and “rest”. Besides, learning the direct relations between numbers is important, because they will decide the order...
Math Word Problem (MWP):
"When a batch of goods is transported, the ratio between the transported goods and ..."

Answer: "100"

Equation: 
\[
\frac{30}{\frac{7}{3+7} - \frac{2}{5}}
\]

Heterogeneous Line Graph Transformer (HLGT) Model

Our major contributions are summarized as follows:

- We transform the math word problems (MWP) into heterogeneous line graph. Dual graph encourages question representation models to discover the implicit meta relationships between word tokens automatically.
- We propose a novel model (HLGT) to enrich question representations. Type-specific parameters and edge dynamic embeddings are adopted to guide message aggregation from neighborhoods and capture the structure information.
- We develop an automated MWP solver based on the HLGT model, and we conducted experiments to demonstrate the effectiveness of our model. Results show that the HLGT model achieves better performance than existing solutions.

2 PRELIMINARIES

In this section, we present the formal problem definition and introduce some basic knowledge of heterogeneous line graph.

Given a math word problem with a sequence of n words \( X = (x_1, x_2, x_3, ... x_n) \), the goal is to generate an equivalent expression tree \( Y = (y_1, y_2, y_3, ... y_m) \). It consists of a set of variables \( y_m \in \{o_1, o_2, o_3, ..., o_l\} \cup \{c_1, c_2, c_3, ..., c_l\} \cup \{q_1, q_2, q_3, ..., q_k\} \), where \( o, c \) and \( q \) stand for operators (e.g., +, -, × and ÷), frequently used constants (e.g., 1, 100, 3.14, etc.) and the Numerical tokens existed in problem statements, respectively. Follow the settings in [41], all the numerical tokens existed in problem X are mapped to a special token "NUM" for we do not care the exact value of these tokens.

Definition 2.1 (Heterogeneous Graph). A heterogeneous graph [36] is defined as a directed graph \( G = (V, E, A, R) \) with various types of nodes and edges, where \( V \) is a complete set of all nodes, \( E \) is a complete set of all edges. And each node \( v \in V \) and each edge \( e \in E \) are associated with their type mapping functions \( \tau(v): V \rightarrow A \) and \( \phi(e): E \rightarrow R \), respectively.

Definition 2.2 (Meta Relation). For an edge \( e = (s, t) \) linked from source node \( s \) to target node \( t \), its meta relation is denoted as \( \tau(s), \phi(e), \tau(t) \). It describes a implicit relation from source node to target node. In this work, we defined 7 node types (i.e., entity, root, unit, rate, fraction, percentage, and other numbers.) and 7 edge types (see Table 1 for details) according to the rule of semantic role labeling [35] and numerical comparison results. In Table 1, \( \tau, r, u, q \) refers to entity, root, rate or unit, and numerical tokens in problem statements, respectively. \( BAE \) and \( LES \) are used to connect two number tokens, and representing \( \geq \) and \( < \) relation respectively. \( DT \) is set for two numbers with different type.

Definition 2.3 (Line Graph). Given an undirected graph \( G_n = (V, E) \), its line graph \( G_L = (V_L, E_L) \) encodes the directed edge adjacency structure of \( G_n \). If \( G_n \) is a node-centered graph, then the line graph \( G_L \) could be viewed as edge-centered graph. For example, for edge \( E(i \rightarrow j) \) and \( E(j \rightarrow k) \) in graph \( G_n \), they also act as two
Solving Math Word Problems Automatically with Heterogeneous Line Graph Transformer for Online Learning

Figure 3: The overall model architecture. Our model has multi-encoders and a tree-based decoder (a). To explicitly incorporate both sequential and structural information, we employ sequence and graph-structure encoders (b). The structure encoder takes the initial representation, heterogeneous SRL graph, and its line graph as inputs to update contextualized representations of nodes and edges. In this process, type and graph structure information are encoded into $h^s_{str}$. Simultaneously, the sequence encoder creates the sequential representation $h^s_{seq}$. We train a biaffine attention module on an auxiliary task (c) to compare numerical values with a comparison loss.

3 PROPOSED MODEL

The proposed method is illustrated in Figure 3. It is inherited from the seq2tree structure and consists of multi-encoders and a tree-based decoder. Specifically, given a math word problem $X$, the word representation initialization module firstly generates initial representations for sequence encoder and structure encoder. Then, these two encoders work in parallel and output embedding vectors with different information. For word order information understanding, sequence encoder adopts a generalized recurrent neural model. For capturing the implicit and complex meta relations between word tokens, we design a heterogeneous line graph transformer (HLGT) module to discover the logical information and encode them into the word representation embedding vectors. In tree-based decoder, the model generates the expression tree $Y$ according to question representations via attention and copy mechanism.

3.1 Words Representation Initialization

We use a two-layer Bidirectional Gated Recurrent Unit (BiGRU) model [8] to encode the word-level representations. Formally, the BiGRU model takes the word sequence of problem statements $X = (x_1, x_2, x_3, ..., x_n)$ as input, and produces a sequence of hidden states $H = (h^x_1, h^x_2, h^x_3, ..., h^x_n)$ as output, where $h^x_i$ and $\overrightarrow{h^x_i}$ represent forward and backward hidden state vectors, respectively:

$$h^x_i = \overrightarrow{h^x_i}; \overleftarrow{h^x_i}.$$

Here, word embedding vectors $E(x_i)$ are obtained via a wording embedding layer. The representations $h^x_i$ has contained semantics

This matrix [6] enables the directed propagation of information on the line graph.
of the word itself and incorporated contextual information, which serves as the basis of problem understanding for sequence and structure encoders:

\[ h_i^L = \overrightarrow{h_i} + h_i^T. \]  

(3)

3.2 Heterogeneous Graph Construction

We refer all number tokens \(X_q\), nouns \(X_n\), units/rates \(X_u\) and predicate verbs\(root\) \(X_r\) as possible nodes of node-centered graph. Next, we define a SRL graph as a subset of nodes in the problem. Specifically, we firstly use string matching to extract all Numbers as we define a SRL graph as a subset of nodes in the problem. Formally, the edges \((r,e)\) is defined according to the annotations of SRL. Next, we identify all the nouns in extracted phrase as entity nodes based on the Part-of-Speech Tagging (PoS) result. At last, if a number token is immediately followed by a unit or rate word, this word will be recognized as a unit or rate-node.

We construct 7 kinds of edges to provide different connections between nodes. Formally, the edges \(E\) label the relationship between the nodes, which correspond to the following four situations:

- **Edge between root and entity**: A directed edge exists between root-node and entity-node in the same SRL section, which is defined according to the annotations of SRL.
- **Edge from entity to entity**: For non-root phrases with more than one node, we select one of the them as the contact node and link it to the root node. If there is a numerical node in this phrase, then this node will be the contact node, otherwise the last occurring entity node will be selected. This edge is defined to connect non-contact and contact nodes in the same role phrase.
- **Edge from unit/rate to number**: For each number and its related unit or rate, a directed edge \(MOD\) is added. Because the unit- or rate-node may contain specific algebraic calculation information (such as * 10 are needed for the conversion of centimeters to decimeters).
- **Edge from number to number**: For node pairs with the same type, an edge containing quantity comparison information would be added (i.e., better and equal \(BAE\), less \(LES\)). For different types of numbers, it is meaningless to compare them, so we use different type \(DT\) edge to connect them. The edges in this situation cluster the same type numbers together and conduct numerical comparison. This edge also serves the function of connecting subgraphs.

3.3 Heterogeneous Line Graph Transformer

Here, we present the details of HLGT module, which combines the advantages of line graph [6] and heterogeneous multi-head attention mechanism [14]. Based on the heterogeneous graph \(G\), HLGT enriches question representations, which collects the meta relation information through multiple iterations of message passing between neighborhoods. For instance in Figure 1, the goal is to calculate how many tons of this batch of goods. The HLGT model is expected to discover the meta path between 30 and origin good through aggregating information from neighbors.

**Origin Graph**. Given the node-centered graph \(G_n\), the output representation \(h_i^L\) of the l-th iteration is computed by equation 4.

\[
\tilde{a}_{ij} = (h_i W_q^{\tau(x_i)})(h_j W_k^{\tau(x_j)}) \odot \psi(z_{ji}),
\]

\[
a_{ij} = \text{softmax}(\tilde{a}_{ij}/\sqrt{d/M}),
\]

\[
\tilde{h}_i = \sum_{m=1}^M \tilde{a}_{ij}(h_j W_o^{\tau(x_j)} \odot \varphi(z_{ji})),
\]

\[
h_i^{l+1} = \text{LayerNorm}(\delta^{\tau(x_i)} h_i + (1 - \delta^{\tau(x_i)}) \tilde{h}_i U_0^{\tau(x_i)}),
\]

where \(\|\) denotes vector concatenation, matrices \(W_q^{\tau(x_i)}, W_k^{\tau(x_j)}\), \(W_o^{\tau(x_i)} \in \mathbb{R}^{d \times d/M}, \psi, \varphi\) and \(U_0^{\tau(x_i)} \in \mathbb{R}^{d \times d}\) are trainable parameters, \(M\) is the number of heads and \(\text{LayerNorm()}\) denotes a normalization layer.

Firstly, \(\tau(x_i)\)-type source node representation and \(\tau(x_j)\)-type target node representation are projected into type-specific Key vector \(W_k^{\tau(x_i)}\) and Query vector \(W_q^{\tau(x_i)}\), respectively. Then, We use heterogeneous mutual attention mechanism to calculate multi-head attention \(a_{ij}\). To maximize parameter sharing while discovering the meta relations between different node pairs, we use \(\psi\) vectors to project edge-features \(z_{ji}\), and then multiply the result with source and target node projection. Next, in message passing part, we also incorporate the dynamic edge features to alleviate the distribution features of implicit meta relations. We replace all edge-type parameters matrix and use dynamic edge representation calculated by the edge-centered module. This way is more flexible than the fixed size parameter matrix to capture the relationship between numbers or between numbers and entities. At last, the node representation needs to be updated over the heterogeneous multi-head attention and type-specific passing message. A type coefficient \(\delta^{\tau(x_i)}\) is adopted to decide how much information is retained from the original representation and how much information aggregated from neighbor nodes. Besides, the passing message \(\tilde{h}_i\) is mapped to type-specific message output vector \(U_0^{\tau(x_i)}\), so that to get the different type distribution.

**Line Graph**. Given edge-centered graph \(G_e\), the updated node representation \(z_{i+1}\) could be calculated with multi-head attention mechanism:
We use Goal-driven Tree Structure (GTS) decoder [41] as our decoder module. Firstly, the final problem representations of numbers can be decomposed further according to the context vector and goal vector. Type-specific parameters help the model to learn implicit relationship between two numbers. We retrieve the vector from a trainable parameter matrix directly. If \( y \) is an operator or a constant, its embedding should be retrieved from a trainable parameter matrix directly. If \( y \) is a number in a problem statement, its embedding is copied from the final representations of problem statements. Due to the decomposition process is implemented as the pre-order operation during the traversal, when predicting sub-goal token, the model will first process the left sub-goal. When it comes to the right sub-goal, the construction of its left sub-tree must have been completed. Then, the embedding of all left sub-tree nodes will be recursively encoded to the root node of the left sub-tree and used in predicting the target of the right sub-tree.

### 3.5 Numerical Comparison

A generalized encoder for MWP should be able to compare the value of two numbers. When we consider how to calculate the difference of two numbers \( q_1 \) and \( q_2 \), the comparative relationship between them can help decide the order they appear in the expression (e.g. \( q_1 - q_2 \) or \( q_2 - q_1 \)). To introduce this inductive bias, we design an auxiliary task to compare the relationship between two numbers. We retrieve all the numerical representation vectors \( h^q_i \) from the final question embeddings. Then, we use a bi-affine [9] binary classifier to determine the comparative relationship between the number pair \((q_1, q_2)\):

\[
\text{BiAffine}(q_1; q_2) = h^q_{x_1} U_i h^q_{x_2}^T + [h^b_{x_1} ; h^b_{x_2}] W_s + b_s, \\
\rho^\text{com}(q_1; q_2) = \sigma(\text{BiAffine}(q_1; q_2)).
\]

Then the ground truth label \( y^\text{com}(q_1; q_2) \) is determined by the relationship between the number pair (e.g., \( > \) is 1 and \( < \) is 0). And the training object can be formulated as

\[
\mathcal{L}^\text{com} = 1/|K^2| \sum_{i=1}^K \sum_{j=1}^K \rho^\text{com}(q_i; q_j),
\]

where \( K \) is the number of numerical tokens in problem.

### 3.6 Training Method

Given a batch of question statements \( X \) and its corresponding ground truth mathematical expression \( Y^g \), the training object \( \mathcal{L}^g \) could be formulated as the sum of negative log-likelihoods of probabilities for predicting expression \( Y^g \). And the final loss function \( \mathcal{L} \) is the weighted sum of \( \mathcal{L}^g \) and \( \mathcal{L}^\text{com} \):

\[
\text{Pr}(Y^g|Y^g, X) = \prod_{t=1}^m \text{Pr}(y_t|Y^g_{t-1}, G, X), \\
\mathcal{L}^g = \sum_{(X, Y^g) \in \mathcal{D}} - \log \text{Pr}(Y^g|Y^g, X), \\
\mathcal{L} = \mathcal{L}^g + \beta^\text{com} \mathcal{L}^\text{com},
\]

where \( y_t \) is the t-th token predicted by the model and \( G \) represents the graph input.
4 EXPERIMENTS

In this section, we evaluate the proposed model in different settings. Ablation study and analysis are also conducted to demonstrate the effectiveness of its components.

4.1 Experimental Settings

Datasets. Math23K [39] is a Chinese dataset that contains 23,162 elementary school level MWP s and corresponding equation expressions and answers. MAWPS [19] is an English dataset with 2,373 math word problems. APE210k [45] is a large-scale dataset with 200,488 training data, 5000 test data and 5,000 valid data. We pre-processed APE210k to ensure that its mathematical expression can be executed, and the execution result is consistent with the answer. After processing, 162,081 training data, 4,046 test data and 4,057 valid data were kept.

Baseline methods. We compared our method with various baselines and state-of-the-art model: T-RNN [38] uses equation normalization method to normalize the duplicated equations and applies seq2seq structure to generate equation expression. GROUP-ATTENTION [22] designs a multi-head attention module with different meanings referring to the transformer mechanism. GTS [41] decomposes the problem into a binary tree composed of sub-goals. Graph2Tree [44] adopts single layer of graph encoder to enrich the number features. MEMD [32] proposes a model with multi-encoders and multi-decoders, and designs a new graph form for math word problem. FECA [45] enhances seq2seq model with external features and copy mechanism. NUMS2T [40] treats all numbers with explicit numerical values and adds external knowledge into encoder model.

Implementation. We utilize PyTorch [26] to implement our method. Specifically, question is first tokenized by Hanlp [12] for English datasets and LTP [3] for Chinese datasets. For the input representation, We initialize the word embedding with the pretrained 300-dimension word vectors. The number of heads М in HLG T is set to 4, and the number of HLG T layer is 2. The hidden size is 512 and the batch size is 64. For optimizer, we use Adam [17] optimizer to minimize loss and set the beginning learning rate as 1e-3. In the training process, the learning rate is halved every 20 epochs for Math23K and 10 for APE210k/MAWPS. The numerical comparison loss coefficient $\beta_{com}$ is set to 0.1. Models are trained in 80 epochs for Math23K, 50 epochs for APE210K, and 30 epochs for MAWPS. During testing, the beam size is set as 5.

4.2 Main Results

The main results are provided in Table 2. Our method achieves excellent expression accuracy of 79.0% for Math23K, 84.8% for MAWPS and 69.8% for APE210K.

From the experimental results: 1) Performance of the models that further process natural language (i.e., Transforming natural language into graph (Graph2Tree, MEMD, ours), introducing external knowledge and dealing numerical token with explicit Values (NUMS2T)) is better compared with the models that simply treat natural language (FECA, GTS) as sequence. This indicates that there is indeed a gap between natural language and machine-readable forms. 2) In addition, Graph2Tree outperforms GTS model, which proves that graph inputs promote the model to learn implicit structure information and improve the model accuracy. 3) Compared with NUMS2T, Our model still shows advantages on Math23K and MAWPS datasets. NUMS2T uses the external knowledge base to extract nodes and edges from problem statements and construct a input graph. However, our model learns hidden meta relationships from SRL graphs automatically. This illustrates that HLG T model could learn meta relations between words without external knowledge.

Although APE210k is the largest dataset, it contains either some unlabeled problems or unexecutable equation expressions. It requires to clean the dataset before training. Due to different processing methods, FECA, NUMS2T and HLG T retain valid set with different number of items.

4.3 Ablation Study

In this section, we investigate the contribution and effectiveness of each components.

Effect of Graph Construction Method. We firstly tested the proposed model on three different heterogeneous graphs:

- Parse Graph + Numerical Comparison Graph [32]: If a word pair has dependency connection, there is an undirected edge between these two words. In addition, for two number tokens $q_i, q_j$, there are two directed edges that represent the quantity comparison relationship in both direction.
- Quantity Cell Graph + Quantity Comparison Graph [44]: This construction uses pre-processing tools to extract verbs, nouns, adjectives and units related to the numbers. Undirected edges are added to the extracted words and the corresponding numerals. Quantity comparison graph is similar to numerical comparison graph, where only a single directed edge stands for the > relationship are retained.

As shown in Table 3, the expression accuracy of the heterogeneous line graph is obviously better than the other two graph construction methods. It may be caused by the following reasons: a) The first construction method lacks the direct connection between
numeral and other word tokens, so the model can not learn the implicit meta relationships between them. b) The second graph construction method only connects numerals with their related words, which leads to the lack of the global context for number representation.

**Effect of Different Components.** We verified the overall performance improvement brought by each module. Here, “HLGT w/o node type” means that the node type matrix were removed from the model. “HLGT w/o line graph” means that we delete line graph edge updating module. “HLGT w/o auxiliary task” means that the final loss function excludes Numerical comparison loss. In addition, we compare HLGT with other graph neural network modules that are good at dealing with the relationship between nodes: 1) Relational-aware schema graph attention network (RGAT) [37] adds edge feature to the formula as an additional term when calculating self-attention; 2) Heterogeneous Graph Transformer (HGT) [14] designs node- and edge-type dependent parameters for model graph heterogeneity.

From Table 4, we could found that: 1) if node-type or line graph are removed, the performance decreases. This consolidates our two motivations that by exploiting the structure of problems and subgraph connections, the model can capturing implicit meta relations between tokens. 2) Without auxiliary task, the expression accuracy are reduced to 78.5% and 84.0% for Math23K and MAWPS datasets respectively, which may be caused by the lack of the ability to compare the numerals. 3) The performance of HGT and RGAT is relatively weak without dynamic edge features captured by line graph module. This verifies that the line graph is helpful for the model to learn the implicit meta relationships. 4) The model with node-type (HLGT, HGT) information performs better than that without node-type (RGAT). It consolidates that node type can be used as a prior knowledge to help the model distinguish different types of nodes.

**Effect of Problem Complexity.** Table 5 shows how well HLGT model performs with the increasing question complexity as compared to best-performing NUMS2T. Follow the experiment in [40], we choose the number of numerical tokens in the problem. The result demonstrates that our proposed HLGT outperforms the baseline at all levels of difficulty. This is because HLGT could learn the meta relations in problems automatically, which could better understand the structure information between word tokens. In addition, HLGT brings more enhancements in complex cases. This may be due to the subgraph connectivity in HLGT model, which provides supplementary information for the inference.
The second-grade students\textsuperscript{1} of Zhenhai School planted trees\textsuperscript{1} on one side of a road\textsuperscript{1}. The students\textsuperscript{2} planted trees every 2 meters. Trees\textsuperscript{3} were planted at both ends of the road\textsuperscript{2}. Finally, they found that a total of 11 trees were planted. How long is the path?

Figure 5: Visualization of meta-relation attention that is trained on node-centric and edge-centric graphs.

4.5 Visualize Graph Attention
To prove that the HLGT model can learn the implicit meta-relation between different nodes automatically, we pick the neighbors that have the largest attention value and plot the meta-relation attention in Figure 5. In this example, we choose number 2 as the target. Among all meta relations, <trees, ARGI, planted, ARGM, 2>, <planted, ARGI, 11, BET, 2> are the two most important meta-relation tracks. At the same time, edge BET aggregates information from neighbor edges in different proportions in line graph module. This proves that HLGT model can automatically learn the meta-relation between neighbor nodes and then decide how much information should be aggregated from others.

5 RELATED WORK

5.1 Math Word Problem
MWP solutions could be categorized into three stages. In the first pioneering stage, the idea of rule-based matching is adopted [25, 34], in which manually-crafted rules plays an important role in the whole system. As a result, these solvers rely heavily on the pre-definition of mathematical expressions template. In the second stage, semantic parsing [11, 18, 21, 29, 33] attracted a lot of research interest. This method focuses on feature engineering, that is, in addition to the original input, the problem statements are transformed into structured input to help the model perform better. Recently, with the increase of training items number, the deep learning method has achieved absolute advantages in Math Word Problem [39]. Since the mathematical expression can be transformed into a symmetric binary tree, the idea of tree-based approaches was proved better performance among deep learning based methods. Seq2SeqET [38] use seq2seq model to directly generate the equivalent expression tree. In addition, equality normalization is adopted to shrink the template space. T-RNN [38] has similar ideas with Seq2SeqET while further improves the method of equality normalization. GTS [41] replaces the sequence decoding model with goal driven decoding, which decomposes the problem into a binary tree composed of sub-goals.

In recent years, the advantages of graph neural network in processing graph structure data gradually appear. And it guides another branch line of MWP: the combination of semantic parsing and seq2seq method. This kind of method usually extracts the key information from problems to form a new graph, and then a graph neural network module on the encoding side. Graph2Tree [44] designs a quantity-centered graph for problem statements and adopts a single layer of graph encoder. MEMD [32] proposes a model with multi-encoders and multi-decoders, and designs a different graph form with Graph2Tree. NUMS2T [40] treats all numbers with explicit numerical values and adds external knowledge graph into encoder model.

5.2 Heterogeneous Graph Neural Networks
To mine relationship between nodes in a heterogeneous graph with multiple types of edges and nodes [36], models should have the ability to handle node- and edge-type information. To maintain node-type spaces, [43] integrates the node-type into the encoder. RGCN [31] and RGAT [37] design edge-type specific linear projection for attention mechanism. HGT [14] fully utilizes the heterogeneity of graphs with node- and edge-type specific parameters matrix. Graph structure is another key point for heterogeneous graph. To address static edge embedding for different graph structure, GTN [2] use sequence model to generate different edge embeddings in different graph structure. Line Graph was adopted in [46] to effectively explore the edge relations in different graph structure.

6 CONCLUSIONS
In this paper, we designed a heterogeneous line graph transformer model to enrich the question representation for automated math word problem solver. To narrow the gap between the problem statements and the machine understandable form, we constructed a heterogeneous graph, over which model could learn meta-relations more directly than unprocessed natural language. We used line graph to discover the implicit meta relations paths between tokens. To help the model distinguish the features contained in different types of nodes, node type-specific linear projection was used in attention calculation. Comprehensive experiments on MWP datasets and showed that the proposed HLGT model captured meta relations in problem statements and achieves good performance.

REFERENCES

[1] Ashton Anderson, Daniel P. Huttenlocher, Jon M. Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In 23rd International World Wide Web Conference, WWW ’14, 687–698.
[2] Deng Cai and Wai Lam. 2020. Graph Transformer for Graph-to-Sequence Learning. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020. 7464–7471.
[3] Wanxiang Che, Yunlong Feng, Libo Qin, and Ting Liu. 2020. N-LTP: A Open-source Neural Chinese Language Technology Platform with Pretrained Models. arXiv preprint arXiv:2009.11616 (2020).
[4] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020. 3438–3445.
[5] Kunlong Chen, Weidi Xu, Xiangyi Cheng, Zou Xiaochuan, Youyi Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020. Question Directed Graph Attention Network for Numerical Reasoning over Text. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020. 6759–6768.
[6] Zhengdaos Chen, Lisha Li, and Joan Bruna. 2019. Supervised Community Detection with Line Graph Neural Networks. In 7th International Conference on
Learning Representations, ICLR 2019.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-Adjusted 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, NAACL-HLT 2019. 2656–2666.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014. 1742–1754.

Timothy Dosat and Christopher D. Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In 5th International Conference on Learning Representations, ICLR 2017.

Dheera Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, Zhaopeng Qiu, Xian Wu, and Wei Fan. 2019. Question Difficulty Prediction for Solving Math Word Problems Automatically with Heterogeneous Line Graph Transformer for Online Learning. Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. 4232–4240.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In WWW ’20: The Web Conference 2020. 2704–2710.

Danning Huang, Jie Liu, Danqing Huang, and Jian Yin. 2018. Neural Math Word Problem Solver with Reinforcement Learning. In Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018. 231–237.

Danning Huang, Shuming Shu, Chen-Yew Liu, and Jian Yin. 2017. Learning Fine-Grained Expressions to Solve Math Word Problems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017. 803–814.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In 3rd International Conference on Learning Representations, ICLR 2015.

Rik Koncel-Kedziorski, Hainanneh Hajishirzi, Ashish Sahbarwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing Algebraic Word Problems into Equations. Trans. Assoc. Comput. Linguistics 3 (2015), 585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hainanneh Hajishirzi. 2016. MAWPS: A Math Word Problem Repository. In NAACL HLT 2016. The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 1152–1157.

Nate Kushman, Luke Zettlemoyer, Regina Barzilay, and Yoav Artzi. 2014. Learning to Automatically Solve Algebra Word Problems. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014. 270–281.

Tom Kwiatkowski, Eunoo Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling Semantic Parsing with On-the-Fly Ontology Matching. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013. 1545–1556.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Zhang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling Intra-Relation in Math Word Problems with Different Functional Multi-Head Attention. In Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019. 6162–6167.

Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, and Shujin Wang. 2021. HMS: A Hierarchical Solver with Dependency-Enhanced Understanding for Math Word Problem. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021. 4232–4240.

Qinzhao Wu, Qi Zhang, Zhiyu Wei, and Xuanjing Huang. 2021. Math Word Problem Solving with Explicit Numerical Values. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021. 5859–5869.

Zhupeng Xie and Shichao Sun. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019. 5399–5405.

Adams Wei Yu, David Doohan, Minli-Thai Luong, Rui Zhao, Kai Chen, Mohamad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In 6th International Conference on Learning Representations, ICLR 2018.

Chuxu Zhang, Dongjin Song, Chao Huang, Amanthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous Graph Neural Network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019. 793–803.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Enhong Chen. 2020. Graph-to-Tree Learning for Solving Math Word Problems. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020. 3982–3997.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Age210K: A Large-Scale and Template-Rich Dataset of Math Word Problems. CoRR abs/2009.11506 (2020).

Yanbin Zhao, Lu Chen, Zhi Chen, Rui Sheng, Caizhan Sun, and Kai Yu. 2020. Line Graph Enhanced AAMT-to-Text Generation with MxOrder Graph Attention Networks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020. 732–741.