Continual Pre-training of Language Models for Math Problem Understanding with Syntax-Aware Memory Network

Zheng Gong[1,4†], **Kun Zhou**[2,4†], **Wayne Xin Zhao**[1,4*], **Jing Sha**[3,5], **Shijin Wang**[5,6], **Ji-Rong Wen**[1,4]

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China [3]iFLYTEK Research, Hefei, Anhui, China
[4]Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China
[5]State Key Laboratory of Cognitive Intelligence, Hefei, Anhui, China
[6]AI Research(Central China), iFLYTEK, Wuhan, Hubei, China

Abstract

In this paper, we study how to continually pre-train language models for improving the understanding of math problems. Specifically, we focus on solving a fundamental challenge in modeling math problems, *i.e.,* how to fuse the semantics of textual description and formulas, which are highly different in essence. To address this issue, we propose a new approach called **COMUS** to **co**ntinually pre-train language models for **m**ath problem **u**nderstanding with **s**yntax-aware memory network. In this approach, we first construct the math syntax graph to model the structural semantic information, by combining the parsing trees of the text and formulas, and then design the syntax-aware memory networks to deeply fuse the features from the graph and text. With the help of syntax relations, we can model the interaction between the token from the text and its semantic-related nodes within the formulas, which is helpful to capture fine-grained semantic correlations between texts and formulas. Besides, we devise three continual pre-training tasks to further align and fuse the representations of the text and math syntax graph. Experimental results on four tasks in the math domain demonstrate the effectiveness of our approach. Our code and data are publicly available at the link: https://github.com/RUCAIBox/COMUS.

1 Introduction

Understanding math problems via automated methods is a desired machine capacity for artificial intelligence assisted learning. Such a capacity is the key to the success of a variety of educational applications, including math problem retrieval (Reusch et al., 2021), problem recommendation (Liu et al., 2018), and problem solving (Huang et al., 2020).

To automatically understand math problems, it is feasible to learn computational representations from problem statement texts with pre-trained language models (PLMs) (Shen et al., 2021; Peng et al., 2021). Pre-trained on the large-scale general corpus, PLMs (Devlin et al., 2019) can be effectively transferred into new domains or tasks by continual pre-training on task-specific datasets. Different from traditional text comprehension tasks, as shown in Figure 1, math problems usually involve a complex mixture of mathematical symbols, logic and formulas, which becomes a barrier to the accurate understanding of math problems.

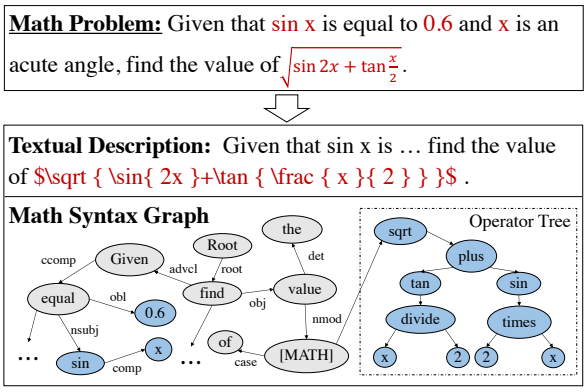


Figure 1: Illustration of a math problem with its textual description and math syntax graph.

However, previous works (Reusch et al., 2021; Shen et al., 2021) mostly oversimplify the issues of math problem understanding. They directly concatenate the formulas with the textual description as an entire sentence, and then perform continual pre-training and encoding without special considerations. Therefore, two major shortcomings are likely to affect the understanding of math problems. First, formulas (the most important elements of the problem) contain complex mathematical logic, and modeling them as plain text may incur the loss of important information. Second, the textual description contains essential explanations or hints about the symbols and logic within the formulas. Hence, it is necessary to accurately capture fine-grained

† Equal contribution. This work was done when the two author were interns at iFLYTEK Research.

* Corresponding author, email: batmanfly@gmail.com

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 5923 - 5933

correlations between words from the description text and symbols from math formulas.

To better model the computational logic of formulas, operator trees are introduced to represent the math formulas (Zanibbi and Blostein, 2012), which are subsequently encoded by graph neural network (GNN). Although these methods can improve the comprehension capacity of math problems to some extent, there still exists a semantic gap between graph encoding and text encoding due to the heterogeneity of formulas and texts. With simple concatenation or self-attention mechanisms (Peng et al., 2021), it is still hard to capture the fine-grained associations among tokens and symbols, *e.g.,* the dependency relation between math symbols and corresponding explanation tokens.

In order to better fuse the information from formulas and texts, our solution is twofold. First, we construct a syntax-aware memory network based on a structure called *math syntax graph* (Figure 1), which integrates operator trees from formulas and syntax trees from texts. The key point lies in that we store the node embeddings from the GNN and dependency relation embeddings as entries of memory networks, and then design the corresponding read and write mechanism, using token embeddings from the PLM as queries. Such a way can effectively associate the representation spaces of the text and formulas. Second, we devise specific continual pre-training tasks to further enhance and fuse the text and graph representations, including the masked language model and dependency triplet completion tasks to improve the understanding of math symbols in the text and formulas logic in the syntax graph, respectively, and the text-graph contrastive learning task to align and unify the representations of the text and graph.

To this end, we propose **COMUS**, to **co**ntinually pre-train language models for **m**ath problem **u**nderstanding with **s**yntax-aware memory network. In our approach, we first encode the textual description and math syntax graph via PLM and GAT, respectively. Then, we add syntax-aware memory networks between the last k layers of PLM and GAT. In each of the last k layers, we first conduct the multi-view read and write operation to fuse the token and node representations, respectively, and then adopt the next layer of PLM and GAT to encode the fused representations. All parameters of our model are initialized from PLMs and will be continually pre-trained by our devised three tasks, namely masked language model, dependency triplet completion and text-graph contrastive learning. Experimental results on four tasks in the math domain have demonstrated the effectiveness of our approach, especially with limited training data.

Our contributions can be summarized as follows:

(1) We construct a novel syntax-aware memory network to capture the fine-grained interactions between the text and formulas.

(2) We design three continual pre-training tasks to further align and fuse the representations of the text and graph data.

(3) Experiments on four tasks in the math domain demonstrate the effectiveness of our model.

2 Preliminaries

In this section, we formulate the problem statement and then introduce the math syntax graph.

Problem Statement. Generally, a math problem consists of a textual description d and several formulas $\{f_1, f_2, \cdots, f_m\}$. The textual description provides necessary background information for the math problem. It is formally denoted as a sequence of tokens $d = \{t_1, t_2, \cdots, t_l\}$, where t_i is either a word token or a mathematical symbol (*e.g.,* a number or an operator). The formulas describe the relationship among mathematical symbols, which is the key to understand and solve the math problem. Each formula consists of a sequence of mathematical symbols, denoted as $f_i = \{s_1, \cdots, s_n\}$.

Based on the above notations, this work focuses on continually pre-training a PLM on unsupervised math problem corpus for domain adaptation. After that, the PLM can be fine-tuned on various tasks in the math domain (*e.g.,* knowledge point classification), and improve the task performance.

Math Syntax Graph. In order to understand the mathematical text and formulas, it needs to capture the complex correlations within words, symbols and operators. Inspired by previous works (Mansouri et al., 2019; Peng et al., 2021), we construct a *syntax graph*, where the textual description is represented as a syntax dependency tree and the formulas are represented as operator trees (OPT).

Specifically, given a math problem consisting of a textual description d and several formulas $\{f_1, f_2, \cdots, f_m\}$, we first utilize the open-source toolkit TangentS[1] to convert each formula into an

[1]https://github.com/BehroozMansouri/TangentCFT

OPT, and Stanza[2] to convert the textual description into a syntax dependency tree. Then, we combine the syntax dependency tree and the OPTs to compose an entire graph, where a special token "[MATH]" is applied to link them. We call such a composite graph as the math syntax graph $\mathcal{G}$ of the math problem. Let $\mathcal{N}$ and $\mathcal{R}$ denote the set of nodes and relations on $\mathcal{G}$, respectively. We can extract dependency triplets from $\mathcal{G}$, where a dependency triplet (h, r, t) denotes that there exists an edge with the relation $r \in \mathcal{R}$ to link the head node $h \in \mathcal{N}$ to the tail node $t \in \mathcal{N}$.

3 Methodology

As shown in Figure 2, our approach aims to effectively encode the textual description and formulas, and fuse these two kinds of information for understanding math problems. In what follows, we first present the base models for encoding math problems, and then introduce the devised syntax-aware memory network and continual pre-training tasks.

3.1 Base Models

Encoding Math Text. We use BERT (Devlin et al., 2019) as the PLM to encode the math text, *i.e.,* the textual description d. Given $d = \{t_1, t_2, \cdots, t_L\}$ of a math problem, the PLM first projects these tokens into corresponding embeddings. Then, a stack of Transformer layers will gradually encode the embeddings to generate the l-th layer representations $\{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \cdots, \mathbf{h}_L^{(l)}\}$. Since the textual description d may contain specific math symbols that were not seen during pre-training, we add them into the vocabulary of the PLM and randomly initialize their token embeddings. These new embeddings will be learned during continual pre-training.

Encoding Math Syntax Graph. We incorporate a graph attention network (GAT) (Veličković et al., 2018) to encode the math syntax graph, which is composed of an embedding layer and a stack of graph attention layers. Given a math syntax graph $\mathcal{G}$ with N nodes, the GAT first maps the nodes into a set of embeddings $\{\mathbf{n}_1, \mathbf{n}_2, \cdots, \mathbf{n}_N\}$. Then each graph attention layer aggregates the neighbors' hidden states using multi-head attentions to update the node representations as:

$$\mathbf{n}_i^{(l+1)} = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}_k^{(l)} \mathbf{n}_j^{(l)}\Big). \quad (1)$$

[2]https://stanfordnlp.github.io/stanza/

where $\mathbf{n}_i^{(l+1)}$ is the representation of the i-th node in the $l+1$ layer, $\Vert$ denotes the concatenation operation, σ denotes the sigmoid function, K is the number of attention heads, $\mathcal{N}_i$ is the set of neighbors of node i in the graph, $\mathbf{W}_k^{(l)}$ is a learnable matrix, and α_{ij}^k is the attention value of the node i to its neighbor j in attention head k.

3.2 Syntax-Aware Memory Network

To improve the semantic interaction and fusion of the representations of math text and the syntax graph, we add k syntax-aware memory networks between the last k layers of PLM and GAT. In the memory network, node embeddings (from the math syntax graph) with dependency relations are considered as slot entries, and we design multi-view read/write operations to allow token embeddings (*e.g.,* explanation tokens or hints) to attend to highly related node embeddings (*e.g.,* math symbols).

Memory Initialization. We construct the memory network based on the dependency triplets and node representations of the math syntax graph. Given the dependency triplets $\{(h, r, t)\}$, we treat the head and relation (h, r) as the key and the tail t as the value, to construct a syntax-aware key-value memory. The representations of the heads and tails are the corresponding node representations from GAT, while the relation representations are randomly initialized and will be optimized by continual pre-training. Finally, we concatenate the representations of heads and relations to compose the representation matrix of *Keys* as $\mathbf{K}^{(l)} = \{[\mathbf{n}_{h_1}^{(l)}; \mathbf{r}_1], [\mathbf{n}_{h_2}^{(l)}; \mathbf{r}_2], \cdots, [\mathbf{n}_{h_N}^{(l)}; \mathbf{r}_N]\}$, and obtain the representation matrix of *Values* as $\mathbf{V}^{(l)} = \{\mathbf{n}_{t_1}^{(l)}, \mathbf{n}_{t_2}^{(l)}, \cdots, \mathbf{n}_{t_N}^{(l)}\}$.

Multi-view Read Operation. We read important semantics within the syntax-aware memory to update the token representations from PLM. Since a token can be related to several nodes within the math syntax graph, we design a multi-view read operation to capture these complex semantic associations. Concretely, via different bilinear transformation matrices $\{\mathbf{W}_1^S, \mathbf{W}_2^S, \cdots, \mathbf{W}_n^S\}$, we first generate multiple similarity matrices $\{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_n\}$ between tokens and keys (head and relation) within the memory, and then aggregate the values (tail) to update the token representations. Given the token representations from the l-th layer of PLM $\mathbf{H}^{(l)} = \{\mathbf{h}_1^{(l)}, \mathbf{h}_2^{(l)}, \cdots, \mathbf{h}_L^{(l)}\}$,

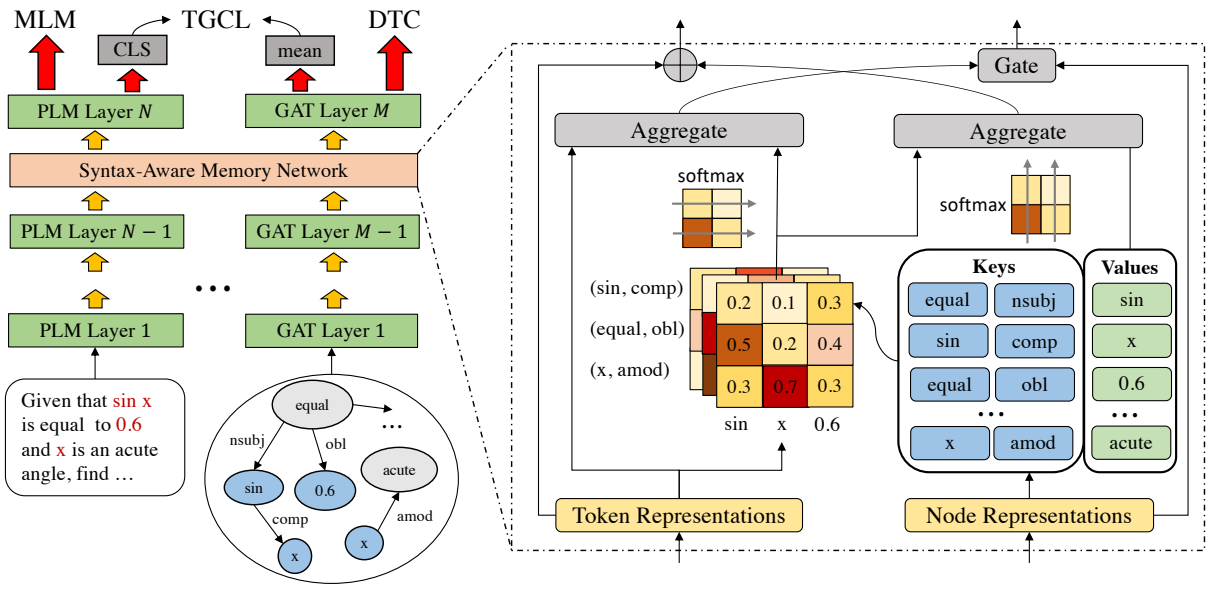


Figure 2: Illustration of our COMUS. We encode the textual description and the math syntax graph using PLM and GAT, respectively, and insert the syntax-aware memory networks in the last k layers to fuse their representations. In the syntax-aware memory network, we utilize the token representations and the node representations as the queries and values, respectively, and implement the *read* and *write* operations to update them.

the similarity matrix $\mathbf{S}_i$ is computed as

$$\mathbf{S}_i = \mathbf{H}^{(l)}\mathbf{W}_i^S{\mathbf{K}^{(l)}}^\top \tag{2}$$

where $\mathbf{W}_i^S$ is a learnable matrix, and an entry $\mathbf{S}_i[j,k]$ denotes the similarity between the j-th token and the k-th key in the i-th view. Based on these similarity matrices, we update the token representations by aggregating the value representations as

$$\hat{\mathbf{H}}^{(l)} = \mathbf{H}^{(l)} + [\alpha_1\mathbf{V}; \alpha_2\mathbf{V}; \cdots; \alpha_h\mathbf{V}]\mathbf{W}^O \tag{3}$$

$$\alpha_i = \texttt{softmax}(\mathbf{S}_i) \tag{4}$$

where $\mathbf{W}^O$ is a learnable matrix and α_i is the attention score distribution along the key dimension. In this way, we can capture the multi-view correlations between tokens and nodes, and the token representations can be enriched by the representations of multiple semantic-related nodes. After that, the updated token representations $\hat{\mathbf{H}}^{(l)}$ are fed into the next layer of PLM, where the Transformer layer can capture the interaction among token representations to fully utilize the fused knowledge from the syntax graph.

Multi-View Write Operation. After updating the token representations, we update the representations of nodes from GAT via memory writing. We still utilize the multi-view similarity matrices $\{\mathbf{S}_1, \mathbf{S}_2, \cdots, \mathbf{S}_h\}$. Concretely, we compute the attention score distribution β using softmax function along the token dimension of the similarity matrices, and then aggregate the token representations as

$$\mathbf{V}_{new}^{(l)} = [\beta_1\mathbf{H}^{(l)}; \beta_2\mathbf{H}^{(l)}; \cdots; \beta_h\mathbf{H}^{(l)}]\mathbf{W}^R \tag{5}$$

$$\beta_i = \texttt{softmax}(\mathbf{S}_i^\top) \tag{6}$$

where $\mathbf{W}^R$ is a learnable matrix. Based on the aggregated token representations, we incorporate a gate to update the representations of the values as

$$z = \sigma(\mathbf{V}_{new}^{(l)}\mathbf{W}^A + \mathbf{V}^{(l)}\mathbf{W}^B) \tag{7}$$

$$\hat{\mathbf{V}}^{(l)} = z \cdot \mathbf{V}_{new}^{(l)} + (1-z) \cdot \mathbf{V}^{(l)} \tag{8}$$

where $\mathbf{W}^A$ and $\mathbf{W}^B$ are learnable matrices. The updated node representations $\hat{\mathbf{V}}^{(l)}$ are also fed into the next layer of GAT, where the graph attention mechanism can further utilize the fused knowledge from the text to aggregate more effective node representations.

3.3 Continual Pre-training

Continual pre-training aims to further enhance and fuse the math text and math syntax graph. To achieve it, we utilize the masked language model and dependency triplet completion tasks to improve the understanding of math text and math syntax graph, respectively, and the text-graph contrastive learning task to align and fuse their representations.

Masked Language Model (MLM). Since the math text contains a number of special math symbols, we utilize the MLM task to learn it for better understanding the math text. Concretely, we randomly select 15% tokens of the input sequence to be masked. Of the selected tokens, 80% are replaced with a special token [MASK], 10% remain unchanged, and 10% are replaced by a token randomly selected from the vocabulary. The objective is to predict the original tokens of the masked ones as:

$$L_{MLM} = \sum_{t_i \in \mathcal{V}_{mask}} -\log p(t_i) \tag{9}$$

where $\mathcal{V}_{mask}$ is the set of masked tokens, and $p(t_i)$ denotes the probability of predicting the original token in the position of t_i.

Dependency Triplet Completion (DTC). In the math syntax graph, the correlation within the dependency triplet (h, r, t) is essential to understand the complex math logic of the math problem. Thus, inspired by TransE (Bordes et al., 2013), we design the dependency triplet completion task to capture the semantic correlation within a triplet. Specifically, for each triplet (h, r, t) within the math syntax graph, we minimize the DTC loss by

$$L_{DTC} = \max\left(\gamma + d(\mathbf{n}_h + \mathbf{r}, \mathbf{n}_t) - d(\mathbf{n}_h + \mathbf{r}^{'}, \mathbf{n}_t), 0\right) \tag{10}$$

where $\gamma > 0$ is a margin hyper-parameter, $d(\cdot)$ is the euclidean distance, and $\mathbf{r}^{'}$ is the randomly sampled negative relation embedding. In this way, the head and relation embeddings can learn to match the semantics of the tail embeddings, which enhances the node and relation representations by capturing the graph structural information.

Text-Graph Contrastive Learning (TGCL). After enhancing the representations of the math text and math syntax graph via MLM and DTC tasks respectively, we further align and unify the two types of representations. The basic idea is to adopt contrastive learning to pull the representations of the text and graph of the same math problem together, and push apart the negative examples. Concretely, given a text-graph pair of a math problem $(d_i, \mathcal{G}_i)$, we utilize the representation of the [CLS] token $\mathbf{h}_i^d$ as the sentence representation of d_i, and the mean pooling of the node representations $\mathbf{n}_i^{\mathcal{G}}$ as the graph representation of $\mathcal{G}_i$. Then, we adopt the cross-entropy contrastive learning objective with in-batch negatives to align the two representations

$$L_{TGCL} = -\log \frac{\exp(f(\mathbf{h}_i^d, \mathbf{n}_i^{\mathcal{G}})/\tau)}{\sum_{i \neq j} \exp(f(\mathbf{h}_i^d, \mathbf{n}_j^{\mathcal{G}})/\tau)} \tag{11}$$

where $f(\cdot)$ is a dot product function and τ denotes a temperature parameter. In this way, the representations of the text and graph can be aligned, and the data representations from one side will be further enhanced by another side.

3.4 Overview and Discussion

Overview. Our approach focuses on continually pre-training PLMs to improve the understanding of math problems. Given the math text and math syntax graph of the math problem, we adopt PLM and GAT to encode them, respectively, and utilize syntax-aware memory networks in the last k layers to fuse the representations of the text and graph. In each of the last k layers, we first initialize the queries and values of the memory network using the representations of tokens and nodes, respectively, then perform the read and write operations to update them using Eq. 3 and Eq. 8. After that, we feed the updated representations into the next layers of PLM and GAT to consolidate the fused knowledge from each other. Based on such an architecture, we adopt MLM, DTC and TGCL tasks to continually pre-train the model parameters using Eq. 9, Eq. 10 and Eq. 11. Finally, for downstream tasks, we fine-tune our model with specific data and objectives, and concatenate the representations of text $\mathbf{h}^d$ and graph $\mathbf{n}^{\mathcal{G}}$ from the last layer for prediction.

Discussion. The key of our approach is to deeply fuse the math text and formula information of the math problem via syntax-aware memory networks and continual pre-training tasks. Recently, MathBERT (Peng et al., 2021) is proposed to continually pre-train BERT in math domain corpus, which applies the self-attention mechanism for the feature interaction of formulas and texts, and learns similar tasks as BERT. As a comparison, we construct the math syntax graph to enrich the formula information and design the syntax-aware memory network to fuse the text and graph information. Via the syntax-aware memory network, the token from math text can trace its related nodes along the relations in the math syntax graph, which can capture the fine-grained correlations between tokens and nodes. Besides, we model the math syntax graph

Task	Train	Dev	Test
KPC	8,721	991	1,985
QRC	10,000	2,000	4,000
QAM	14,000	2,000	4,000
SQR	250,000	11,463	56,349

Table 1: Statistics of the datasets.

via GAT, and devise the DTC task to improve the associations within triplets from the graph, and the TGCL task to align the representations of the graph and text. In this way, we can better capture graph structural information and fuse it with textual information. It is beneficial for understanding logical semantics from formulas of math problems .

4 Experiment

4.1 Experimental Setup

We conduct experiments on four tasks in the math domain to verify the effectiveness of our approach.

Pre-training Corpus. Our pre-training corpus is collected from a Chinese educational website Zhixue [3], which consists of 1,030,429 problems of high school math exams and tests. Each math problem contains the information of problem statement, answer and solution analysis. For data preprocessing, we first transform these collected problems from the HTML format into plain text format, then extract and convert the formulas and mathematical symbols into a unified LaTex mathematical format.

Evaluation Tasks. We construct four tasks based on the collected math problems for high school students, which cover math problem classification and recommendation. The statistics of these tasks are summarized in Table 1.

• **Knowledge Point Classification** (KPC) is a multi-class classification task. Given a math question, the goal is to classify what knowledge point (KP) this question is associated with. The knowledge points are defined and annotated by professionals, and we finally have 387 KPs in this task.

• **Question-Answer Matching** (QAM) is a binary classification task to predict whether an answer is matched with a question. For each question, we randomly sample an answer from other problems as the negative example.

• **Question Relation Classification** (QRC) is a 6-class classification task. Given a pair of math questions, this task aims to predict their relation (*e.g.,* equivalent, similar, problem variant, conditional variant, situation variant, irrelevant).

• **Similar Question Recommendation** (SQR) is a ranking task. Given a question, this task aims to rank retrieved candidate questions by the similarity.

Evaluation Metrics. For classification tasks (KPC, QRC, QAM), we adopt Accuracy and F1-macro as the evaluation metrics. For the recommendation task (SQR), we employ top-k Hit Ratio (HR@k) and top-k Normalized Discounted Cumulative Gain (NDCG@k) for evaluation. Since the length of candidate list is usually between 6 and 15, we report results on HR@3 and NDCG@3.

Baseline Methods. We compare our proposed approach with the following nine baseline methods:

• **TextCNN** (Kim, 2014) is a classic text classification model using CNN on top of word vectors.

• **TextRCNN** (Lai et al., 2015) combines both RNN and CNN for text classification tasks.

• **GAT** (Veličković et al., 2018) utilizes the attention mechanism to aggregate neighbors' representations to produce representation for each node.

• **R-GCN** (Schlichtkrull et al., 2018) extended Graph Convolutional Network with multi-edge encoding to aggregate neighbors' representations.

• **BERT-Base** (Devlin et al., 2019) is a popular pre-trained model. We use the bert-base-chinese, and add some new tokens into the original vocab to represent specific symbols in math problem dataset.

• **DAPT-BERT** (Gururangan et al., 2020) continually pre-trains BERT on the domain-related corpus. We use our collected math problem dataset with the masked language model task for implementation.

• **BERT+GAT** concatenates the [CLS] embedding from BERT and mean node embedding from GAT as the representation of a math question.

• **DAPT-BERT+GAT** replaces BERT in BERT+GAT with the DAPT-BERT.

• **MathBert** (Peng et al., 2021) continually pretrain BERT on the math corpus with similar pretraining tasks, and revises the self-attention layers for encoding the OPT of formulas.

Implementation Details. For baseline models, all hyper-parameters are set following the suggestions from the original papers. For all PLM-related models, we implement them based on HuggingFace Transformers [4] (Wolf et al., 2020). For the models

[3] http://www.zhixue.com

[4] https://huggingface.co/transformers/

Tasks	KPC		QAM		QRC		SQR	
Metrics	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro	HR@3	NDCG@3
TextCNN	51.2	31.7	91.6	91.6	75.1	55.8	0.321	0.301
TextRCNN	56.8	40.3	89.3	89.2	80.3	62.9	0.334	0.317
GAT	42.5	28.5	90.0	89.9	66.6	45.4	0.315	0.300
R-GCN	40.7	26.0	91.6	91.5	70.4	50.0	0.316	0.298
BERT-Base	59.4	36.0	96.8	96.8	82.3	63.1	0.578	0.576
BERT+GAT	61.1	38.0	97.0	96.9	83.0	64.3	0.568	0.566
DAPT-BERT	67.1	45.2	98.8	98.7	85.9	67.7	0.641	0.643
DAPT-BERT+GAT	67.8	47.3	98.9	98.9	85.8	67.2	0.646	0.649
MathBert	66.4	43.2	98.9	98.9	86.4	68.3	0.640	0.641
COMUS	**72.6**	**57.9**	**99.5**	**99.5**	**88.9**	**81.4**	**0.658**	**0.660**

Table 2: Main results on four downstream tasks. The best and the second best methods are marked in bold and underlined fonts respectively.

Tasks	KPC							
Ratio	40%		20%		10%		5%	
Method	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro
DAPT-BERT	53.1	27.9	38.6	15.2	26.4	7.7	16.8	4.2
DAPT-BERT+GAT	53.3	27.5	38.3	15.5	26.2	6.8	11.8	2.5
MathBERT	49.6	32.1	31.2	11.1	19.5	5.7	8.4	1.9
COMUS	**62.7**	**41.5**	**52.2**	**27.8**	**36.9**	**15.0**	**22.1**	**7.1**
Tasks	**QRC**							
Ratio	40%		20%		10%		5%	
Method	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro	Accuracy	F1-macro
DAPT-BERT	78.8	59.7	73.5	52.7	65.5	46.1	61.4	40.3
DAPT-BERT+GAT	81.4	62.3	73.3	53.1	69.1	48.5	61.8	38.4
MathBERT	80.5	60.9	73.3	47.9	65.6	38.3	58.0	22.6
COMUS	**82.6**	**67.4**	**77.7**	**57.1**	**69.8**	**49.6**	**64.6**	**40.7**

Table 3: Performance comparison *w.r.t.* different amount of training data on KPC and QRC tasks.

combining PLM and GAT, we set GAT's number of layer, attention head and hidden states as 6, 12 and 64, respectively. And we set the number of syntax-aware memory network layers k as 2 for our proposed COMUS.

In the continual pre-training stage, we initialize the weights of all models with bert-base-chinese [5] and pre-train them on our pre-training corpus with the same hyper-parameter setting as follows. We continually pre-train the parameters with a total of 128 batch size for 100,000 steps. And the max length of input sequences is set as 512. We use AdamW (Loshchilov and Hutter, 2019) optimization with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and apply learning rate warmup over the first 5% steps, and linear decay of the learning rate. The learning rate is set as $1e^{-4}$. We set τ as 0.07 for our TGCL tasks. It costs about 40 hours to perform the continual pre-training on 4 Tesla-V100-PCIE-32G GPUs.

During fine-tuning on downstream tasks, we use AdamW with the same setting as pre-training. And batch size for all experiments is set as 32. The learning rate is set to $3e^{-5}$ for pre-training based methods, and $1e^{-3}$ for other methods.

[5]https://huggingface.co/bert-base-chinese

4.2 Main Results

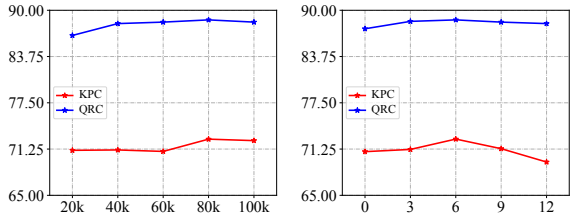
The results of all the comparison methods on four tasks are shown in Table 2. Based on these results, we can find:

As for non-pre-training methods, text-based methods (*i.e.,* TextCNN and TextRCNN) outperform GNN-based methods (*i.e.,* GAT and R-GCN). It indicates that text representations are more capable of understanding math problems than graph representations in our dataset. Overall, non-pre-training methods perform worse than pre-training based methods, since pre-training based models have learned sufficient general knowledge during the pre-training on large-scale corpus.

Among the five pre-training methods, we can have two major findings. First, combining PLMs with GNN yields performance improvement in most cases. The reason is that GNN can capture the structural semantics from formulas as the auxiliary information to help PLMs understand the math problem, but the improvement is unstable, since these methods simply concatenate the representations of the text and graph without deeply fusing them. Second, continual pre-training brings a significant improvement on all the evaluation

	KPC		QRC	
Method	Acc	F1	Acc	F1
COMUS	**72.6**	**57.9**	**88.9**	**81.4**
- w/o GAT	69.4	49.2	87.9	78.3
- w/o $BERT$	41.7	27.2	64.1	39.6
- w/o $Memory$	69.4	49.2	88.1	73.7
- w/o MLM	36.5	21.9	70.2	51.2
- w/o DTC	70.8	55.3	87.8	73.5
- w/o $TGCL$	71.9	56.5	87.9	69.8

Table 4: Ablation study of our approach on the KPC and QRC tasks.



(a) Pre-training Steps (b) GAT Layers

Figure 3: Performance comparison *w.r.t.* the number of pre-training steps and GAT layers

tasks. General-purpose PLMs can't effectively understand mathematical semantics, and it is the key to adapt them to the math domain via continual pre-training.

Finally, by comparing our approach with all the baselines, it is clear to see that our model performs consistently better than them on four tasks. We utilize the syntax-aware memory network to fuse and interact the representations of textual descriptions and formulas, and adopt three continual pre-training tasks to further align and enhance these representations. Among these results, we can see that our model achieves a large improvement on the KPC task. A possible reason is that it requires a deeper semantic fusion of formulas and text for identifying the correct knowledge points.

4.3 Few-shot Learning

To validate the reliability of our method under the data scarcity scenarios, we conduct few-shot experiments on KPC and QRC tasks by using different proportions of the training data, *i.e.,* 5%, 10%, 20% and 40%. We compare our model with DAPT-BERT, DAPT-BERT+GAT and MathBERT.

Table 3 shows the evaluation results with different ratios of training data. We can see that the performance substantially drops when the size of training set is reduced. However, our model performs consistently better than the others across different tasks and metrics. It demonstrates that our model is capable of leveraging the data more effectively with the help of the syntax-aware memory networks and continual pre-training tasks. With 5% training data, our model exceeds the best baseline by a large margin. It further indicates that our model is more robust to the data scarcity problem.

4.4 Ablation Study

Our proposed approach contains several complementary modules and pre-training tasks. Thus, we conduct experiments on KPC and QRC tasks to verify the contribution of these modules and tasks. Concretely, we remove the module GAT, BERT, Syntax-Aware Memory Network, or the task MLM, DTC and TGCL, respectively.

In Table 4, we can see that the performance drops by removing any modules or pre-training tasks. It shows the effectiveness of these modules or pre-training tasks in our proposed model. Especially, the model performance significantly decreases when we removing the textual encoder BERT, which implies that the text representations are more important for math problem understanding. Besides, we can see that removing MLM also results in a large performance drop, since it is the key pre-training task for our text encoder.

4.5 Hyper-Parameters Analysis

Our proposed model contains a few parameters to tune. In this part, we tune two parameters and examine their robustness on model performance, *i.e.,* the number of GAT Layer and the continual pre-training steps. We conduct experiments on KPC and QRC tasks and show the change curves of Accuracy in Figure 3.

We can observe that our model achieves the best performance in 80k steps. It indicates that our model can be improved by continual pre-training gradually and may overfit after 80k steps. Besides, our model achieves the best performance with 6 GAT layers, which shows that 6 GAT layers are sufficient to capture the information in syntax graph.

5 Related Work

In this section, we review the related work from the following two aspects, namely math problem understanding and continual pre-training of language models.

Math Problem Understanding. Math problem understanding tasks focus on understanding the texts, formulas and symbols in math domain. A

surge of works aim to understand the math formulas for problem solving or mathematical information retrieval. In a typical way, the formula is usually transformed as a tree or graph (*e.g.,* Operator Tree (Zanibbi and Blostein, 2012)), then network embedding method (Mansouri et al., 2019) and graph neural network (Song and Chen, 2021) are utilized to encode it. Besides, a number of works focus on understanding math problem based on the textual information. Among them, Math Word Problem (MWP) Solving is a popular task that generates executable mathematical expression for the math word problem to produce the final answer. Numerous deep learning based methods have been proposed to tackle the MWP task, including Seq2Seq (Chiang and Chen, 2019; Li et al., 2019), Seq2Tree (Wang et al., 2019; Qin et al., 2020), and Pre-trained Language Models (Kim et al., 2020; Liang et al., 2021). More recently, several studies attempt to model more complex math problems (Huang et al., 2020; Hendrycks et al., 2021) that require a deep understanding of both textual and formula semantics.

Continual Pre-training of Language Models. Continually pre-training can effectively improve pre-trained model's performance on new domains or downstream tasks (Gururangan et al., 2020). To achieve it, most of previous works either continually optimize the model parameters with BERT-like tasks on domain or task related corpus (*e.g.,* scientific (Beltagy et al., 2019) and bio-media (Lee et al., 2020)), or design new pre-training objectives for task adaption (*e.g.,* commonsense reasoning (Zhou et al., 2021) and dialogue adaption (Li et al., 2020)). Besides, several works (Wang et al., 2020; Xiang et al., 2020) utilize both domain-related corpus and new pre-training objectives for continual pre-training, or revise the Transformer structure of PLMs for better adaption (Ghosal et al., 2020). For math problem understanding, the recently proposed MathBERT (Peng et al., 2021) adopts math domain corpus and formula-related pre-training tasks for continual pre-training.

6 Conclusion and Future Work

In this paper, we proposed COMUS, a continual pre-training approach for math problem understanding. By integrating the formulas with the syntax tree of mathematical text, we constructed the math syntax graph and designed the syntax-aware memory network to fuse the semantic information from the text and formulas. In the memory network, we treated tokens from the text and triplets from the graph as the queries and slot entries, respectively, and modeled the semantic interaction between tokens and their semantic-related nodes via multi-view read and write operations. Besides, we devised three continual pre-training tasks to further enhance and align the representations of the textual description and math syntax graph of the math problem. Experimental results have shown that our approach outperforms several competitive baselines on four tasks in the math domain.

In future work, we will consider applying our method to solve more difficult math-related tasks, *e.g.,* automatic math problem solving and analysis generation. Besides, we will also consider incorporating external math domain knowledge into our model to improve the understanding of mathematical logic and numerical reasoning.

Ethical Consideration

In this part, we discuss the main ethical consideration of this work: (1) Privacy. The data adopted in this work (*i.e.,* pre-training corpus and fine-tuning data) is created by human annotation for research purposes, and should not cause privacy issues. (2) Potential Problems. PLMs have been shown to capture certain biases from their pre-trained data (Bender et al., 2021). There are increasing efforts to address this problem in the community (Ross et al., 2021).

Acknowledgement

This work was partially supported by Beijing Natural Science Foundation under Grant No. 4222027, and National Natural Science Foundation of China under Grant No. 61872369, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, the Outstanding Innovative Talents Cultivation Funded Programs 2021 and Public Computing Cloud, Renmin University of China. This work is also supported by Beijing Academy of Artificial Intelligence (BAAI). Xin Zhao is the corresponding author.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the*

9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2656–2668.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Deepanway Ghosal, Devamanyu Hazarika, Abhinaba Roy, Navonil Majumder, Rada Mihalcea, and Soujanya Poria. 2020. KinGDOM: Knowledge-Guided DOMain Adaptation for Sentiment Analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3198–3210, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset.

Zhenya Huang, Qi Liu, Weibo Gao, Jinze Wu, Yu Yin, Hao Wang, and 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*, pages 1729–1732.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3768–3779.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167.

Junlong Li, Zhuosheng Zhang, Hai Zhao, Xi Zhou, and Xiang Zhou. 2020. Task-specific objectives of pre-trained language models for dialogue adaptation. *arXiv preprint arXiv:2009.04984*.

Zhenwen Liang, Jipeng Zhang, Jie Shao, and Xiangliang Zhang. 2021. Mwp-bert: A strong baseline for math word problems. *arXiv preprint arXiv:2107.13435*.

Qi Liu, Zai Huang, Zhenya Huang, Chuanren Liu, Enhong Chen, Yu Su, and Guoping Hu. 2018. Finding similar exercises in online education systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1821–1830.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Behrooz Mansouri, Shaurya Rohatgi, Douglas W Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. 2019. Tangent-cft: An embedding model for mathematical formulas. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*, pages 11–18.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3780–3789.

Anja Reusch, Maik Thiele, and Wolfgang Lehner. 2021. Tu_dbs in the arqmath lab 2021, clef.

Candace Ross, Boris Katz, and Andrei Barbu. 2021. Measuring social biases in grounded vision and language embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 998–1008, Online. Association for Computational Linguistics.

Michael Sejr Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2021. Mathbert: A pre-trained language model for general nlp tasks in mathematics education. *arXiv preprint arXiv:2106.07340*.

Yujin Song and Xiaoyu Chen. 2021. Searching for mathematical formulas based on graph representation learning. In *International Conference on Intelligent Computer Mathematics*, pages 137–152. Springer.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.

Weiran Wang, Qingming Tang, and Karen Livescu. 2020. Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6889–6893. IEEE.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Suncheng Xiang, Yuzhuo Fu, Guanjie You, and Ting Liu. 2020. Unsupervised domain adaptation through synthesis for person re-identification. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE.

Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(4):331–357.

Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, and Xiang Ren. 2021. Pre-training text-to-text transformers for concept-centric common sense. In *International Conference on Learning Representations*.