scientific reports

Check for updates

OPEN

A machine solution for math word problems based on semantic understanding enhancement

Yanli Wang[1], Ming Yan[2]✉, Pengpeng Jian[2], Yangrui Yang[2] & Yang Li[2]

The adaptive understanding of problem text with various semantics is challenging for machines when solving math word problems. This challenge is particularly important in the context of intelligent technology promoting equitable and sustainable development in education. Most existing methods focus on numerical information processing and ignore important semantic information, which leads to limited improvement of solution accuracy. Therefore, this study proposes a machine solution based on semantic understanding enhancement. First, a knowledge-enhanced pre-trained language model is constructed as a semantic encoder to integrate background information such as phrases and entities to enhance the understanding of lexical, syntactic, and semantic issues. Then, pooling operations are introduced to improve the semantic understanding further, and a readable binary expression tree is generated using a tree structure decoder. Finally, a judgment mechanism based on confidence is proposed to ensure the accuracy of the solution and improve the training efficiency. The experiments show that this method is superior to other baselines on both Chinese and English datasets, which proves its effectiveness and feasibility. This result not only provides new ideas and methods for mathematical solutions but also creates new possibilities for the combination of intelligence and sustainable education.

Keywords Math word problems, Machine solution, Pre-trained language model, Semantic enhancement, Pooling, Confidence

In recent years, intelligent education has gained wide attention with the development of artificial intelligence and the spread of sustainable concepts. As an important part of intelligent education, machine solutions have been widely used, greatly promoting the sustainable utilization of educational resources.

With the popularization of machine solutions, students can obtain rich learning resources and accurate solution services through online platforms, whether in the city or the country. Thus, the problem of uneven regional educational resources has been effectively alleviated, and the balanced distribution of educational resources has been realized. In addition, machine solutions can also help reduce the cost of education and achieve the sustainable use of educational resources. At the same time, the continuous updating and upgrading of machine solutions can promote the innovation and development of education patterns, and it can inject new impetus into the sustainable development of education. The authors of this paper focus on the study of solving elementary Math Word Problems (MWPs), which are basic and challenging subproblems of machine solving mathematical problems.

The solutions to MWPs are mainly based on deep learning methods, there are two categories. The first category comprises methods based on traditional language models. Sequence models based on neural networks, such as RNNs (Recurrent Neural Networks) and LSTM (Long Short-Term Memory), and reinforcement learning models, such as DRL (Deep Reinforcement Learning). They are common natural language models and are designed to perform language tasks such as understanding, classification[1], and generation[2]. For example, in DNS[3], an RNN-based Seq2Seq (sequence-to-sequence) model is integrated for application to MWPs for the first time, solving the difficulty of traditional methods needing to capture features manually. In RecursiveNN[4], a template-solving method based on RNN is integrated and expressions are deduced.

The second category comprises methods based on pre-trained language models. Pre-trained language models (PLMs) have received extensive attention in NLP (Natural Language Processing). The main idea is that the model undergoes large-scale unsupervised learning and is fine-tuned in various downstream tasks. The most famous models are BERT[5], GPT[6], XLNet[7], etc. Their excellence has been demonstrated in text classification,

[1]Henan University of Economics and Law, Zhengzhou 450016, Henan, China. [2]School of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou 450046, Henan, China. ✉email: yanming@stu.ncwu.edu.cn

sentiment analysis[1], generative tasks such as dialogues and summaries[2,8], and other language tasks. Inspired by these tasks, PLMs have also been used for MWP solving. For example, Li et al.[9] believed that the current methods lack an understanding of the MWP-solving mode and added contrast learning to construct BERT-CL. They used a semantic encoder to aggregate problems with similar prototype equations to improve the efficiency of problem-solving. The authors of[10] believed that MWP solving has additional numerical reasoning requirements, and they proposed a set of arithmetic-enhanced pre-training tasks considering logical reasoning and numerical properties for MWP-BERT, which effectively improved the model's performance. Due to their excellent semantic representation ability, PLMs are better suited to solving the difficulty of problem understanding than traditional language models, so we chose this method to solve MWPs.

The MWP solution process is divided into problem understanding, inference prediction, and answer generation. Among them, problem understanding is the foundation. Previous methods based on PLMs either seek to combine other learning mechanisms or pay too much attention to numerical representation, ignoring the importance of problem understanding. As shown in Table 1, when the semantics of the question are relatively complex, the model often predicts the wrong equation and answer because it cannot understand the exact meaning of the text. To solve the semantic understanding bias in MWPs, we reconstruct the BERT model to obtain a new knowledge-enhanced pre-trained model named KnBERT (Knowledge-BERT)[11]and its three-stage knowledge masking strategy and new semantic pre-training tasks effectively enhance its understanding of problem texts.

With the cross-penetration of technologies between different fields, many image-based processing methods have been transferred to NLP tasks and achieved satisfactory results. Yoon[14] used a Convolutional Neural Network (CNN) for sentence classification and made four transformations for comparison. In ConS2S[15]CNN is implemented to replace the common RNN for learning and training. To solve the semantic representation problem of MWPs and enhance the understanding of texts, we attempt to add the mean pooling of CNN for MWP solving. Through a series of average operations, the pooling can extract comprehensive semantic features of the text. The solver can obtain a more comprehensive understanding of the problem text based on KnBERT.

Recent advances in PLMs have demonstrated their excellence in MWPs. However, due to the possibility of generating groundless answers[16,17], users cannot always accept the results of reasoning unconditionally. Therefore, we design a judgment mechanism based on confidence. After a binary expression tree is predicted, it is not solved first; instead, the confidence of the expression is compared with a threshold. If it is lower than the threshold, no further solution operation is performed, and the solution is directly judged to have failed. This mechanism improves the training efficiency of the model while ensuring the accuracy of the solution.

The main contributions of this study are as follows:

- We design a knowledge-enhanced pre-trained language model named KnBERT. Compared with the existing knowledge-enhanced language models, the innovation of KnBERT lies in the introduction of a three-stage masking strategy and pre-training tasks. This design effectively alleviates the semantic understanding deviation of traditional models when dealing with mathematical reasoning problems (MWPs), and pays more attention to the in-depth understanding at the semantic level in the process of knowledge enhancement.
- We construct a novel MWP solver named KnBERT-TD. This uses the language model KnBERT as a semantic encoder, and it uses a tree structure as a decoder, which ensures the uniqueness and legitimacy of its solution equations. The solver effectively improves the accuracy of the solution.
- We introduce the mean pooling of CNNs, which is widely used in computer vision. By gathering all the semantic information in a sentence area, the semantic representation of the text is improved, and the understanding of the problem is further enhanced.
- We propose a judgment mechanism based on confidence. A threshold is set for confidence to judge whether the solution equation is trustworthy. This mechanism significantly enhances the model's training efficiency while maintaining the accuracy of the solution.

The rest of the paper is organized as follows. In "Related work" section presents research-related work. In "KnBERT Pre-trained Model" section designs a pre-trained language model is designed to solve the semantic

Problem: Xiaoming read a storybook, read 1/6 of the book on the first day, and read 24 pages on the second day, the number of pages read on the third day is 150% of the total number of the first two days, and there is still a quarter of the book did not read. How many pages are in this book?	
Correct Solution Equation: x=(24 + 24*150%)/(1-(1/6)-(1/6)*150%-(1/4))	
Correct Answer: 180	
DNS: x =((1–1/6) + 24)/1.5+(1–1/4)/ (1–1/6)/2.5	Answer: 16
GTS[12]: x = 24*1.5/(1-(1/6)-(1/6)*1.5-(1/4))	Answer: 108
BERT-CL: x= (1-(1/6)*1.5-(1/4))*(24 + 24*1.5)	Answer: 30
MWP-BERT: x= (1-(1/6)*2.5-(1/4))*(24*1.5)	Answer: 12
Graph2Tree[13]: x = 24*2.5/(1-(1/6)-(1/6)*2.5-(1/4)	Answer: 360
KnBERT-TD(Ours): x=(24*2.5*(4/3))/(1–1/6*2.5*(4/3)	Answer: 180

Table 1. The results of different models for solving complex problems.

understanding bias. In " Methodology" section proposes an MWP solver, KnBERT-TD, for improving accuracy and a judgment mechanism based on confidence for enhancing efficiency. In the "Experiments and Results" section, experiments and analysis are performed. In "Conclusions", a summary of this paper is presented. In the "Discussion" section, the prospect of future research is discussed.

Related work

Automatic machine solution

The machine solution aims to develop intelligent systems that can automatically understand and answer various problems. The history of machine solutions can be traced back to the 1950s. Alan Turing, the father of artificial intelligence, proposed the classic Turing test[18] to confirm whether computers have "intelligence", thereby laying a solid theoretical foundation for later generations of AI and machine solutions. In 1964, Bobrow[19] developed the STUDENT system, regarded as the first AI answering system.

In the 21st century, machine solutions ended the long settling period. Geometric Problem Solution (GPS) in the field of mathematics has taken the lead in terms of rapid development. Wu et al.[20–22] have made outstanding contributions to GPS and geometric proof problems. Since then, methods for geometrically related problems have been continuously developed. In 2017, Xu[23] proposed a method for the automatic addition of geometric auxiliary lines based on a Monte Carlo tree search. He summarized the auxiliary line addition method to build a cognitive model and applied a knowledge derivation network to solve it. Gan[24] built a syntactic-semantic hybrid model for geometric language, and it extracted and understood the relationships in geometric problems.

Math word problem solution

As an important branch of machine solutions, MWPs have been obtained using a variety of methods—from the rule-based method[25] to the statistical learning method[26] and the semantic analysis method[27]. However, these methods require a large number of manual operations, and the accuracy achieved is not satisfactory. Thus, they have been gradually replaced by new technologies.

With the rise of machine solutions and deep learning, they are also being used to solve MWPs. For example, in 2016, Liang[28] proposed a machine-solving method based on label statistics; this method converts problem text information into a logical structure based on labels, and the problem can be solved after the annotation of semantic information. In 2017, Wang et al. input problem text into a model, and a solution equation was output through the reasoning of the neural network; this solves the difficulty of traditional methods needing to capture features manually, as well as greatly improving efficiency and accuracy. After that, Xie et al. proposed a goal-driven mechanism based on a tree decoder; this mechanism decomposes the target into multiple sub-targets. Zhang et al. designed a quantity unit graph and a quantity comparison graph to capture the relationship and order between the values in a problem and to obtain a more accurate solution. In MathDQN[29], the reward mechanism of deep reinforcement learning was implemented to predict answers.

In recent years, pre-trained language models have achieved great performance in language tasks and are gradually being used to solve MWPs. In EPT[30], an enhanced ALBERT model[31] was implemented as an encoder. In mBERT-LSTM[32], multilingual BERT was implemented to study cross-language and multilingual mathematical problems. MWP-BERT was reconstructed according to the features of MWPs. REAL[33] was the first to combine analog learning with pre-trained models, which emphasized analog learning rather than template-based learning.

KnBERT pre-trained model

Aiming to solve the problem of the semantic understanding deviation of MWPs, we reconstruct a knowledge-enhanced pre-training model named KnBERT (Knowledge-BERT). By using a three-stage knowledge masking strategy and semantic pre-training tasks, the model can study the implicit relationship; common sense; and background knowledge, such as morphology, grammar, and semantics. This semantic knowledge will help the model understand the problem correctly and make correct reasoning. We now introduce KnBERT in detail.

Knowledge masking strategy

BERT learns semantic associations between contexts by randomly masking and predicting words in text. Although this basic masking enables the model to judge the smoothness of the statement, it fails to deeply understand its exact meaning. To make up for this deficiency, we design a new masking strategy for KnBERT. By integrating the information of phrases and entities into the vector, the model can implicitly learn knowledge such as entity relations and entity attributes. Figure 1 shows the masking strategy of the three stages.

Phrase-level masking

The first stage of KnBERT is a basic masking strategy similar to that of BERT, so it is unnecessary to go into detail. In the second stage, KnBERT focuses on the phrase masking in the sentence. In English text, lexical analyses and segmentation tools are used to define its boundaries, while language-specific segmentation tools are used to extract phrase information in Chinese text. At this stage, KnBERT not only takes the basic language unit as the training input but also performs random masking and prediction for certain phrases in the sentence. Finally, KnBERT integrates the acquired phrase knowledge into the text word embedding.

Entity -level masking

Named entities such as personal, institutional, and place names become the focus at this stage. These entities in the text often carry the relationship information of variables, which are very important for extracting the implicit relationship of the sentence. Similar to the second stage, the model first identifies and analyzes the

Fig. 1. Three-stage masking strategy.

named entities in the sentence, and then it masks and predicts them. Later, the phrase and entity knowledge are integrated into the word embedding. A text vector representation containing rich semantics is finally generated after careful processing by the encoder.

Pre-training tasks

KnBERT designs three pre-training tasks for words, structures, and semantics. They capture lexical, grammatical, and semantic information, respectively. The specific tasks are as follows:

Word-related pre-training tasks

Capitalization Prediction (CP): This is mainly designed for English problems. Considering the special meaning of capitalized words in English, combined with the advantages of BERT's branch model, this task is used to find semantic connections between special words.

Keyword Prediction (KP): Certain words may appear repeatedly in different clauses of the same question. They are usually common or relevant to the answer, known as "keywords". By improving the ability of the model to capture keywords, it deepens the understanding of the text content.

Structure-related pre-training tasks

Sentence Reordering (SR): This divides the problem text into several clauses randomly and regroups them to generate a new question. SR aims to improve the ability of the model to grasp the relation in clauses.

Sentence Position Judge (SPJ): The numbers "0" and "1" indicate that the positions of two sentences are adjacent and non-adjacent in one question. The number "2" indicates that two sentences are in different question texts. Sentences that are close to each other have more correlation, while sentences that are far away have less correlation. The model determines the degree of correlation between sentences according to the distance, and this deepens the understanding of the text structure.

Semantically related pre-training tasks

Discourse Relation Prediction (DRP): The dataset proposed by Sileo[34] is used to train KnBERT, and the semantic understanding ability is enhanced by labeling the discourse relations between sentences and finding sentence pairs with a strong semantic correlation. At the same time, corresponding training resources are prepared for Chinese to ensure the ability of the model across languages.

Methodology

KnBERT-TD

Problem text is taken as the input, and the generated expression is taken as the output[35]. Therefore, we adopt the encoder-decoder structure of the Transformer to construct the solver model. Figure 2 shows the complete flow of the KnBERT-TD. First, the problem text is transformed into the corresponding word embedding state. KnBERT acts as a semantic encoder to extract a semantic-rich hidden state vector through continuous knowledge learning and merging after the input of the word embedding state. Then, a mean pooling is added between the encoder and decoder. A mean representation vector containing all the semantic features of the text is obtained following a sequence of averaging operations. However, considering that Seq2Seq will generate equations that cannot be calculated, such as "13 + 21) *5", and the issue of repeatability, such as "x = 4 + 7 + 2–3" and "x = 4 + 7 – 3 + 2", the tree acts as the decoder to predict the generation probability of the binary tree node. Finally, an equation corresponding to the problem is obtained by traversing the expression tree in the pre-order. The working principle of each part of the solver is described below.

Encoder

The problem text sequence $X = \{x_1, x_2, \ldots, x_n\}$ is preprocessed before entering the encoder. Next, the encoder embedding matrix M_{sen} converts the segmentation mark of sequence X into the corresponding word embedding w_i and obtains the word embedding sequence $W = \{w_1, w_2, \ldots, w_n\}$. The hidden state at a certain moment is acquired by the state in two directions, which include the previous moment and the next moment. The encoder calculates the word embedding w_s of time s and the hidden state $\overrightarrow{h_{s-1}}$ of the last time to obtain the forward $\overrightarrow{h_s}$:

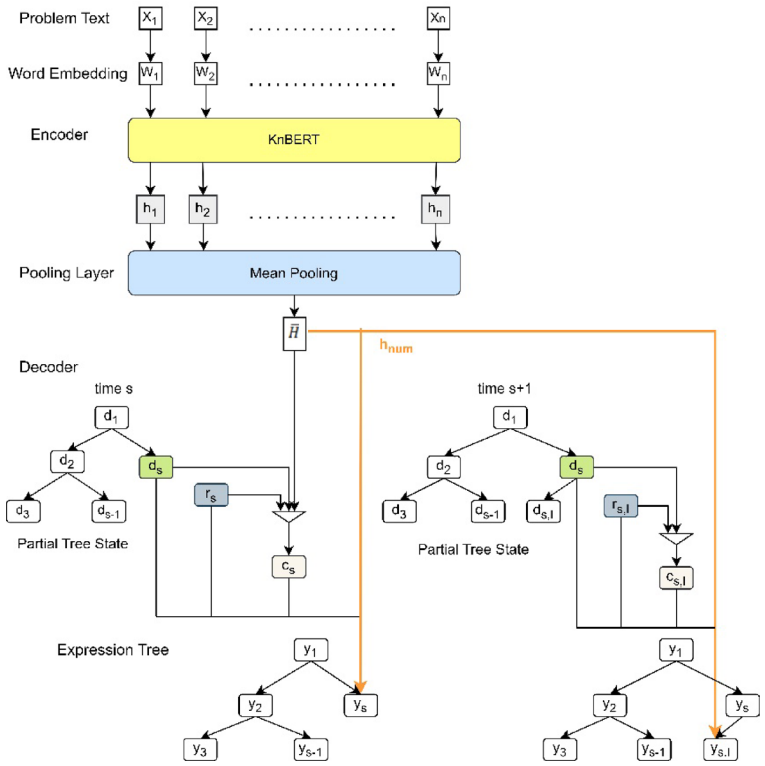

Fig. 2. The complete structure of KnBERT-TD.

Input: Problem text sequence $X = \{x_1, x_2, \ldots, x_n\}$
Output: Extracted text hidden state sequence $H = \{h_1, h_2, \ldots, h_n\}$

1. Problem text preprocessing:
2. Get the direct relation group from the relation extraction algorithm
3. Extract direct relation location, label using '$[CLS]$' and '$[END]$'
4. Add '$[CLS]$' to the beginning of sequences $X = \{[CLS], x_1, \cdots, x_n\}$
5. Add '$[END]$' to the end of sequences $X = \{[CLS], x_1, \cdots, x_n, [END]\}$
6. Get $X' = \{[CLS], x'_1, \cdots, x'_n, [END]\}$ by one-hot encoding
7. Put X' into embedding matrix $\boldsymbol{M_{sen}}$
8. Get a word embedding sequence $W = \{w_1, w_2, \ldots, w_n\}$ containing direct relation group
9. Get $\overrightarrow{h_s}$ from $Encoder\left(w_s, \overrightarrow{h_{s-1}}\right)$
10. Get $\overleftarrow{h_s}$ from $Encoder\left(w_s, \overleftarrow{h_{s+1}}\right)$
11. Obtain the final hidden state h_s by combining $\overrightarrow{h_s}$ + $\overleftarrow{h_s}$
12. Return $H = \{h_1, h_2, \ldots, h_n\}$

Algorithm 1. Preprocessing and encoding process.

$$\overrightarrow{h_s} = Encoder\left(w_s, \overrightarrow{h_{s-1}}\right) \tag{1}$$

The reverse hidden state $\overleftarrow{h_s}$ is calculated using the word embedding w_s and the hidden state $\overleftarrow{h_{s+1}}$:

$$\overleftarrow{h_s} = Encoder\left(w_s, \overleftarrow{h_{s+1}}\right) \tag{2}$$

Finally, the hidden state h_s at time s is obtained by merging the two directions:

$$h_s = \overrightarrow{h_s} + \overleftarrow{h_s} \tag{3}$$

We also add a mean pooling layer behind the encoder. The hidden sequence $H = \{h_1, h_2, \ldots, h_n\}$ is pooled to obtain an average vector $\bar{H}$, which is put into the decoder. We describe mean pooling later.

Mean pooling

Considering the excellent performance of pooling in computer vision, it should be applied to MWP solving. This would have two benefits: Firstly, the pooling layer would facilitate parameter matrix compression and narrow the dimensionality of multidimensional data, thereby alleviating model workload and accelerating computational speed while ensuring the preservation of essential data features during dimensionality reduction. Secondly and most importantly, the semantic feature would be extracted deeply through a reduced parameter matrix, and a hidden representation vector containing a comprehensive understanding of the problem would be obtained, thereby improving the semantic representation of the text and further enhancing the understanding of the problem.

Max pooling and mean pooling are two common pooling operations. The principle of max pooling is to choose the maximum value in the corresponding region of a sentence as the pooling value of the sentence vector. It may result in the loss of some semantic information. Mean pooling takes the average value of a region as the pooling value of the corresponding sentence. It brings together all the state information of the text and promotes the understanding of the solution model as far as possible. Therefore, we choose mean pooling to deal with hidden state vectors.

Mean pooling transforms the state representation of every problem text into a 4*4 feature map and utilizes a 2*2 filter to "scan" in two steps. The details are shown in Fig. 3. The choice of a 4*4 feature map is not a random decision, but rather is determined by the following factors. Firstly, in terms of the balance between dimensionality reduction and information retention, this study aims to retain the core features of the input data as much as possible while reducing dimensionality. The 4*4 feature map maintains sufficient information while avoiding excessive compression, ensuring the integrity of semantic features. This choice is not arbitrary but the result of experimentation and optimization. Secondly, from the perspective of the trade-off between computational efficiency and model performance, choosing smaller feature maps (such as 4*4) helps to reduce the consumption of computational resources. In the MWP inference scenario, if the feature map is too small (such as 2*2), it may cause the model to lose its ability to understand long or complex texts. If it is too large (such as 6*6), the pooling may not have a sufficient effect to simplify the problem. To verify the effectiveness of the pooling method, this study also set up experiments to compare feature maps of different sizes (2*2, 4*4, 6*6) and different types of pooling methods (Max pooling and mean pooling). For details, please refer to the ablation experiment section. The selection of 4*4 feature maps is mainly to balance performance and computational efficiency, ensuring model performance while avoiding unnecessary computational overhead.

The vector values within the region are averaged, and the average is output to the next pooling layer. This calculation process is repeated until the final vector value can no longer be averaged. The final values of each sentence are grouped as the average representation vector $\bar{H}$. The calculation formula is as follows:

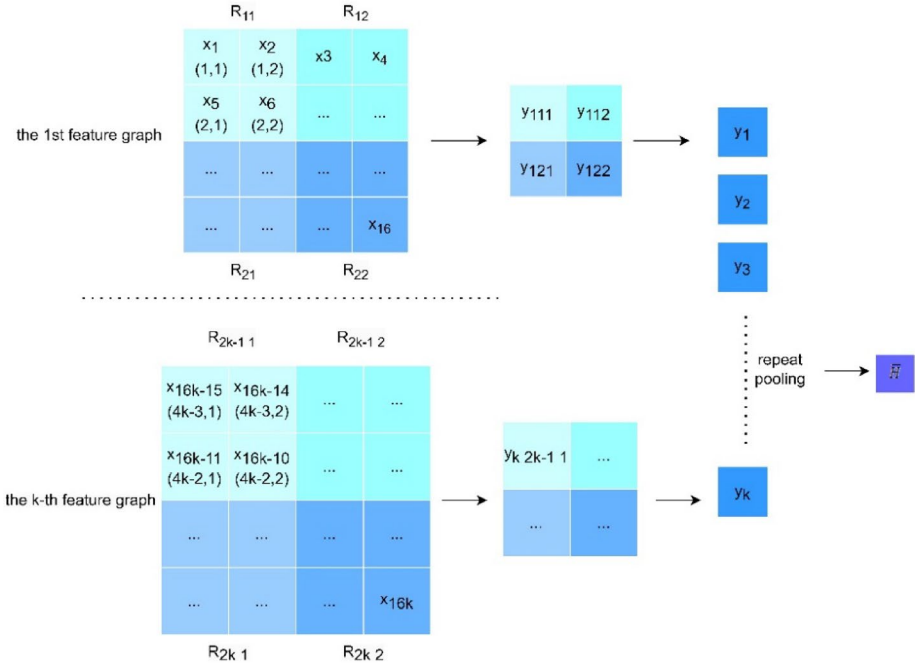

Fig. 3. The "scan" process of mean pooling.

Input: Problem text average representation vector $\bar{H}$, word embedding state sequence W
Output: Binary expression sequence Y

1. Put $\bar{H}$, sequence W into decoder
2. For y_n in $\{y_1, \cdots, y_n\}$:
3. Obtain the context vector c_s based on average presentation vector and word embedding states by the attention mechanism
4. Obtain the status representation d_s at time s based on node embedding $e(y_{s-1})$ and the context state r_s by the attention mechanism
5. Push head nodes to the stack
6. Recursively compute $d_{s,l}$ and $d_{s,r}$
7. Push child nodes to the stack
8. For $P(y_s|y_{<s}, X)$ in $\{P(y_1), P(y_2) \cdots, P(y_n)\}$:
9. The normalized function Softmax is used to obtain the probability $P_{op}(y_s) = softmax(\boldsymbol{W_{op}}[d_s; \boldsymbol{c_s}; r_s])$ that the current node is an operator
10. The normalized function Softmax is used to obtain the probability $P_{num}(y_s) = softmax(\boldsymbol{W_{num}}[d_s; \boldsymbol{c_s}; r_s; h_{num}])$ that the current node is a number
11. Obtain the final generation probability of the node by $argmax = \left((1-\beta_t)P_{op}(y_s), \beta_t P_{num}(y_s)\right)$
12. End for
13. End for
14. **Return** $Y = \{y_1, y_2, \cdots, y_s, \ldots, y_n\}$

Algorithm 2. Decoding processing.

$$y_{kij} = \frac{1}{|R_{ij}|} \sum_{(p,q) \in R_{ij}} x_{kpq} \tag{4}$$

Where y_{kij} represents the output values of rectangle R_{ij} with the k-th feature graph, x_{kpq} represents the element at $(p,\ q)$ in R_{ij}, and $|R_{ij}|$ is the number of elements in rectangle R_{ij}.

The tree structure decoder

If node y_s generated at time s is an operator, this means that it is an intermediate node and must generate its left child $y_{s,l}$ and right child $y_{s,r}$. If y_s is a number, it would be a leaf node. After all the nodes are predicted, the solver will convert the binary tree into the solving equation, which is calculated by the compiler. The prediction process of the nodes is as follows:

The average representation vector and the word embedding sequence are used as the input of the decoder; thus, the context vector c_s can be obtained. The state of the decoder at different times is calculated according to the vector. The formulas are as follows:

$$c_s = Decoder\left(\bar{H},\ W\right) \tag{5}$$

$$d_{s,l} = \sigma\left(W_{left}\left[d_s : c_s : r_s : e\left(y_s\right)\right]\right) \tag{6}$$

$$d_{s,r} = \sigma\left(W_{right}\left[d_s : c_s : r_s : e\left(y_s\right)\right]\right) \tag{7}$$

Where W_{left} and W_{right} are the weight matrices, σ is the sigmoid function, $e\left(y_s\right)$ is the embedding state of the $s-th$ binary node, c_s is the context vector of the encoder's hidden state, r_s is the context state of the partial expression generated at the last moment, d_s is the state representation at time s, and $d_{s,l}$ and $d_{s,r}$ are the left child state and the right child state of d_s. The children are calculated recursively from the top and all states of the tree nodes at time s can be acquired. Root node y_1 uses $\bar{H}$ to initialize d_1.

The determination of whether the currently generated node is an operator or a number needs to be made through a probability calculation. The decoding states are acquired after the above operations, and the generation probability of the node can be calculated:

$$P_{op}\left(y_s\right) = soft\max\left(W_{op}\left[d_s : c_s : r_s\right]\right) \tag{8}$$

$$P_{num}\left(y_s\right) = soft\max\left(W_{num}\left[d_s : c_s : r_s : h_{num}\right]\right) \tag{9}$$

$$\beta_t = \sigma\left(W_z\left[d_s : c_s : r_s : h_{num}\right]\right) \tag{10}$$

$$P(y_s|y_{<s}, X) = \begin{cases} (1-\beta_t)\, P_{op}\left(y_s\right) \\ \beta_t P_{num}\left(y_s\right) \end{cases} \tag{11}$$

Where W_{op} and W_{num} are the weight matrices, $\beta_t \in [0,1]$ is a gated value that decides whether to generate an operator or a number, and $y_{<s}$ represents the node generated before time s. The final generation probability $P(y_s|y_{<s}, X)$ is determined by the operator probability $P_{op}(y_s)$ and the numerical probability $P_{num}(y_s)$. By predicting nodes one by one, a complete binary solution tree is finally generated.

Judgment mechanism

Although language models are trained extensively before performing specific tasks, they are still machines in essence and inevitably produce flawed reasoning. In the ideal case, it would be possible to have the model learn the confidence measure directly for each prediction. However, this has proven to be a big challenge. There is no underlying truth label that can be used for confidence estimation in most machine learning tasks[36]. Instead of learning confidence directly from traditional labels, we design a method to motivate neural networks to generate confidence estimates during training. After predicting the tree expression, the confidence of each question is compared to the threshold. If it is lower than the threshold, no further solution operation is performed, and the solution is directly judged to have failed. This mechanism can not only guarantee solving accuracy but can also shorten the model training time and improve training efficiency.

Confidence estimation

In a test situation, a good strategy for students to optimize their score is to answer all the questions that they are confident about without using prompts and to ask for prompts for uncertain questions to increase accuracy. At the end of the test, the number of prompts can be seen as an approximate indicator of their confidence for each question. This strategy can be applied to a neural network model that can learn and estimate the confidence of the model without relying on the true label.

To give the neural network model the ability to request a prompt, we add a confidence branch parallel to the prediction branch in the feedforward architecture. The confidence branch is shown in Fig. 4. A confidence branch consists of one or more fully connected layers, the last of which outputs a single scalar between 0 and 1 (parameterized to sigmoid). The confidence value, c, represents the degree of confidence that the model can produce correct predictions given the input to the problem. If the model is confident that it can generate the correct expression, its output c value will be close to 1; otherwise, its output c will be close to 0.

Before normalization, the model accepts the problem input X to produce a prediction logit and a confidence logit. For the prediction logit, the softmax function is used to obtain the prediction probability p for each node of the binary tree. The confidence logit is passed by sigmoid to obtain the confidence estimate c.

$$p, c = f(X, \Theta) \tag{12}$$

$$p_i, c \in [0,1], \sum_{i=1}^{M} p_i = 1 \tag{13}$$

To "tip" the model during training, the prediction probability of the softmax function is adjusted by interpolating between the original binary tree prediction and the target probability distribution Y, where the degree of interpolation is expressed by the confidence of the network:

$$p_i^{'} = c * p_i + (1 - c) * Y_i \tag{14}$$

Figure 4 illustrates this visually. The modified prediction probability is used to calculate the expression loss as usual. For the loss calculation, we take the negative logarithmic likelihood function:

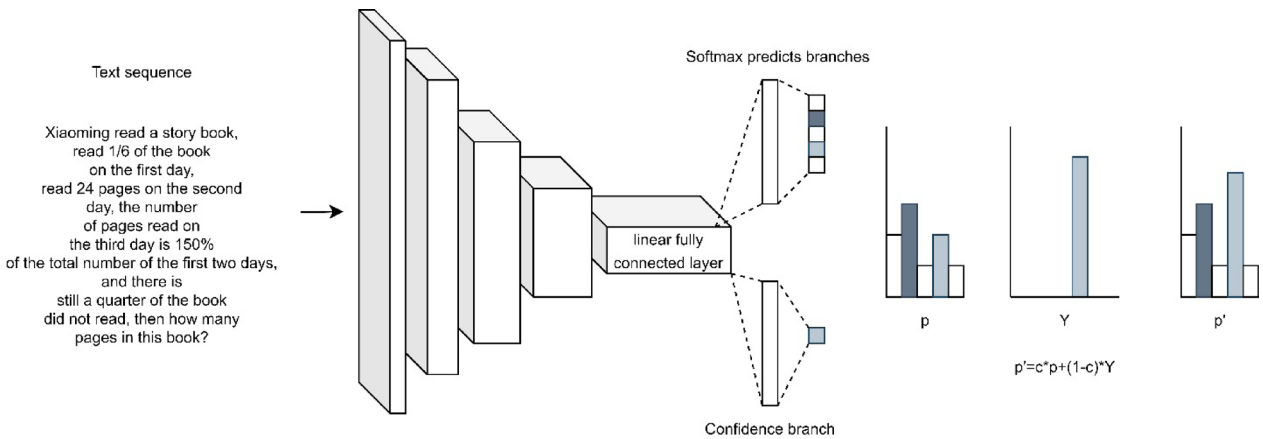

Fig. 4. Confidence evaluation branch.

$$\mathcal{L}_e = -\sum_{i=1}^{M} \log\left(p_i^{'}\right) Y_i \tag{15}$$

To prevent the solver model from always choosing $c = 0$ and accepting the entire base truth value to minimize the expression loss, a logarithmic penalty is added to the loss function, called the confidence loss:

$$\mathcal{L}_c = -\log(c) \tag{16}$$

Therefore, the final loss of the solver model is simply the sum of the expression loss and the confidence loss. The confidence loss is weighted by the hyperparameter λ, which balances the expression loss and confidence loss:

$$L = \mathcal{L}_e + \lambda \mathcal{L}_c \tag{17}$$

Parameter optimization

As the training goes on, c often converges to the unity of all samples. This may cause learned confidence estimates to lose their usefulness. To ensure that confidence estimates remain meaningful in the training (i.e., $c \to 1$ corresponds to the sample problem that is predicted correctly, and $c \to 0$ corresponds to the sample problem that is predicted incorrectly), a budget hyperparameter, β, is introduced, which represents the amount of confidence penalties that the network is allowed to generate. In the process of training, the confidence loss weight λ is adjusted after each weight update so that the confidence loss is tilted toward β. If $\mathcal{L}_c > \beta$, then λ is increased (the network is not confident, increasing the cost of requesting a prompt). If $\mathcal{L}_c < \beta$, then λ is reduced (the network is confident, reducing the cost of requesting a prompt). It was observed that selecting a reasonable β value (between 0.1 and 1.0) did not affect the model's performance.

Confidence judgment

Each time the model predicts the answer to a question, it generates a confidence value c for that result. The higher the confidence value, the more confident the model is in its prediction. A smaller confidence value indicates that the result is untrustworthy. In the pre-training stage, the confidence of each question was estimated, and the scatter diagram of the confidence distribution was drawn through regression analysis, as shown in Fig. 5. As can be seen from the figure, the confidence of the initial training model is generally low, and more wrong predictions are generated. This is because the model just contacts the solving task; it is not familiar with the solving process, the understanding ability of the problem is weak, and the reasoning ability has not reached its peak. With the deepening of the learning process, the model automatically learns to master the solution method, the confidence value becomes higher and higher, and the confidence of correct prediction and wrong prediction tends to their respective interval ranges. It can be seen that there is a threshold δ of confidence value, below which the prediction is wrong, and above which the result is guaranteed to be correct (there are also individual wrong results, which are unpredictable). For the expression predicted by the decoder, it is not solved first, but the confidence of the expression is compared with the threshold value. If it is lower than the threshold value, the subsequent solving operation is no longer carried out, and the solution is directly determined to fail. This strategy not only ensures accuracy but also improves the model training efficiency. Therefore, in this mechanism, the selection of the threshold is extremely crucial, directly affecting the accuracy and efficiency of the model. For this reason, this study proposes a dynamic threshold adjustment mechanism. During the training phase, by analyzing the confidence distribution of each question, the 95th percentile of the confidence in the dataset is used as the initial threshold. This threshold means that the model will only further solve the problem when the confidence level of the problem exceeds this value. This method can effectively eliminate uncertain predictions and ensure the quality of the solution. As the model continues to learn during the training process, the confidence distribution gradually changes, and a dynamic threshold adjustment mechanism is designed.

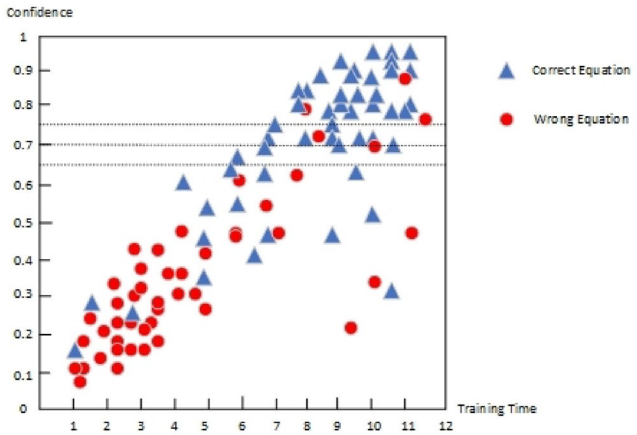

Fig. 5. Scatter plot of confidence distribution.

Each time the training round reaches a certain stage, the threshold will be dynamically adjusted based on the performance of the training set to adapt to the gradually increasing confidence of the model.

Experiments and results

The solving precision of KnBERT-TD is compared with that of other baselines on Chinese and English MWPs. To verify the effectiveness of each strategy, ablation experiments are also conducted. The set strategies are removed one by one, and the performance of the model has decreased to some extent. This also proves the importance of the synergistic effect between various strategies in improving the overall performance of the model when dealing with Chinese and English MWPs. These experimental results provide a firm foundation for the further optimization of the model.

Pre-training

For Chinese MWPs, the Ape-210k dataset is used to pre-train the model. Ape-210k[37] is a large-scale Chinese dataset with multiple and complex mathematical problems. It requires not only a comprehension of natural language but also common-sense knowledge. For English MWPs, pre-training is performed through MathQA's training set. MathQA[38] is an English dataset including geometry, probability, and other more difficult problems.

Configuration

Datasets and evaluation metric

For Chinese, the Math23k and Ape-210k datasets are used to evaluate model performance. For English, MathQA and MAWPS are used. In addition, the accuracy of the answer serves as a metric for evaluating the model's performance.

Parameter details

The code implementation of this experiment is based on the PyTorch framework, and all model training processes are completed on this platform. In terms of model configuration, we first set the number of training epochs in the pre-training to 150 to ensure that the model can fully learn domain-specific corpus features. For the formal training (fine-tuning) phase, 125 epochs were set up to optimize the pre-trained model and adapt it to the target task.

During the Chinese training, the batch size is set to 64. The initial learning rate of the Adam optimizer is set to 3e-5, which is slightly lower and aims to adjust the model's parameters more finely. In terms of regularization, a Dropout rate of 0.5 was set to prevent overfitting of the model. The size of Beam Search is 5. By controlling the size of Beam Search, the decoding efficiency and result quality are balanced, ensuring the stability and accuracy of the model during the inference stage.

In English training, the batch size is set to 16. The initial learning rate of the Adam optimizer is set to 0.0001. The dropout rate is 0.1. The size of Beam Search is 5.

Baselines

Classical baselines based on traditional language models

DNS: The text of the problem is input into the model, and a solution equation is output after the reasoning of the neural network, which solves the difficulty of traditional methods needing to capture features manually.

Math-EN[39]: By calculating the generation probabilities of three different models, the result with the highest probability serves as the solution. In addition, a normalization technique is proposed to address the issue of equation repeatability.

RecursiveNN: A template-solving method is presented based on RNNs. A Seq2Seq model is used to predict the template and deduce an expression.

GTS: A goal-driven mechanism is proposed to generate an expression tree via target decomposition through layers of subtrees.

TSN-MD[40]: The teacher-student network is constructed with multiple decoders to generate a diverse range of equivalent equations and increase the diversity and generalization of the solution.

Graph2Tree: A quantity unit graph and a quantity comparison graph are designed to capture the relationship and order in the numbers of the problem, making the solution more accurate.

Baselines based on PLMs

BERT-CL: Li et al. believe that the current methods lack an understanding of the solution mode, so they combined BERT with contrast learning to construct BERT-CL. The semantic encoder BERT aggregates problems with similar prototype equations and separates dissimilar ones to improve the efficiency of the solution.

REAL: REAL emphasizes analogical learning rather than traditional template-based learning.

BERTGen and RoBERTaGen[41]: Lan et al. tested the problem-solving ability of BERT and RoBERTa on different datasets when researching 17 MWP-solving models. They were found to be more accurate than most models without a PLM.

GPT-4[42]: GPT-4 is a large-scale multimodal model improved based on GPT3.5, which can accept image and text inputs and generate text outputs. Can effectively complete natural language reasoning and generation tasks. Although GPT-4 may not perform as well as humans in many real-world scenarios, it demonstrates human-level performance on various professional and academic benchmarks.

SBI-RAG[43]: A pattern-based instruction retrieval enhanced generation (SBI RAG) framework has been developed, which includes a large language model that emphasizes step-by-step reasoning to guide solution generation by utilizing patterns. And introduce the "reasoning score" indicator to evaluate the quality of the solution. Improved the clarity of reasoning and facilitated a more structured problem-solving process.

Baselines		Accuracy			
		Math23K	Ape-210k	MathQA	MAWPS*
Classical baselines	DNS	–	66.2%	–	59.5%
	Math-EN	66.7%	–	–	69.2%
	RecursiveNN	68.7%	–	–	–
	GTS	75.6%	73.2%	71.3%	82.6%
	TSN-MD	77.4%	–	–	–
	Graph2Tree	77.4%	–	72.0%	83.7%
	BERTGen	76.6%	75.4%	–	86.9%
Baselines based on PLM	RoBERTaGen	76.9%	74.2%	–	88.4%
	REAL	82.3%	–	–	–
	BERT-CL	83.2%	81.5%	76.3%	–
	GPT-4	84.3%	83.0%	77.3%	–
	SBI-RAG	80.5%	–	–	–
	LCR	80.1%	–	–	–
	KnBERT-TD	85.7%	83.5%	77.9%	89.0%

Table 2. Comparison of answer accuracy between KnBERT-TD and baseline models. * indicates five-fold cross-validation.

Baselines		Recall				F1 score			
		Math23k		Ape-210k		Math23k		Ape-210k	
		Arithm-etic	Equation	Ape-unsolvable	Ape-clean	Arithme-tic	Equation	Ape-unsolvable	Ape-clean
Classical baselines	DNS	–	–	–	–	–	–	–	–
	Math-EN	–	–	–	–	–	–	–	–
	RecursiveNN	–	–	–	–	–	–	–	–
	GTS	–	–	–	–	–	–	–	
	TSN-MD	–	–	–	–	–	–	–	–
	Graph2Tree	–	–	–	–	–	–	–	–
Baselines based on PLM	BERTGen	–	–	–	–	–	–	–	–
	RoBERTaGen	–	–	–	–	–	–	–	–
	REAL	–	–	–	–	–	–	–	–
	BERT-CL	81.3%	79.1%	70.3%	73.6%	0.822	0.811	0.755	0.773
	GPT-4	82.5%	80.0%	72.7%	74.0%	0.834	0.821	0.775	0.782
	SBI-RAG	–	–	–	–	–	–	–	–
	LCR	–	–	–	–	–	–	–	–
	KnBERT-TD	83.6%	82.7%	78.9%	81.2%	0.846	0.842	0.811	0.823

Table 3. Comparison of recall and F1 score between KnBERT-TD and baseline models.

LCR[44]: The first attempt was made to use retrieval-enhanced generation to solve mathematical problems, proposing a new method for measuring mathematical logical similarity and designing an automatic filtering mechanism to construct a set of reference problems that combine semantic and logical similarity. By using carefully crafted positive and negative example prompts, guide the model to adopt sound reasoning logic.

Main results

The experimental results are shown in Tables 2 and 3. KnBERT-TD achieved the highest accuracy on all datasets.

Through the discussion of the experimental results, we can get the following three points:

(1) A tree decoder improves model performance. In the first type of baseline, except for DNS, Math-EN, and RecursiveNN, the others have a tree structure, and their solution accuracies reach more than 70%. In the PLM baselines, BERT-CL utilizes a tree structure for decoding; this is also more accurate than REAL, SBI-RAG, LCR, BERTGen, and RoBERTaGen, which do not have a tree structure.
(2) Pre-trained language models perform better in language tasks. Models such as SBI-RAG, LCR, REAL, and BERT-CL have achieved over 80% accuracy due to their optimization on specific tasks and advanced inference mechanisms, demonstrating their advantages in specific scenarios. In contrast, the performance of BERTGen and RoBERTaGen is slightly inferior, with an accuracy of less than 80%, because they only fine-tune the pre-trained model without conducting in-depth optimization for specific tasks. Therefore, these lower results, to some extent, demonstrate the limitations of the method. Among all baseline models,

Items	Math23K		MathQA	
	w pre-training	w/o pre-training	w pre-training	w/o pre-training
Accuracy	85.7%	75%	77.9%	70.3%

Table 4. Effect of pre-training on accuracy.

Items	+basic-level masking	+ phrase-level masking	+entity-level masking (KnBERT-TD)
Accuracy	81.3%	82.9%	85.7%

Table 5. The effect of masking strategies on accuracy.

GPT-4 performs the best, which can be attributed to its more complex architecture and training methods as the most advanced large language model currently available. This makes GPT-4 more capable in language generation and understanding, enabling more precise task execution. However, due to the implementation of more precise optimization strategies for MWP in this method, KnBERT TD performs slightly better than GPT-4 and performs the best among all baseline models. The performance differences of these models highlight the importance of task-specific design and optimization, while also reflecting the potential of larger models in a wider range of application areas. With the advancement of technology, there may be more innovative models and methods emerging in the future, further driving the development of natural language processing.

(3) The semantic understanding enhancement method can effectively solve the semantic understanding bias of MWPs and improve the solving accuracy of the model. KnBERT-TD achieves the highest accuracy compared to all baselines. This is due to the pre-training tasks and masking strategies employed by KnBERT, which enable it to acquire the common-sense knowledge present in the original corpus and the background information hidden within the context. In addition, mean pooling also further improves the semantic representation of the text so that the solver can obtain more accurate answers.

(4) KnBERT-TD has a stronger reasoning ability for solving Math Word Problems. From the results in Table 3, it can be seen that the recall rates of all models in the slightly simpler Math23k arithmetic problems and Ape-clean are higher than those in the Math23k equation problems and Ape-unsolvable. Moreover, the overall value is that Ape210k is less than Math23k (because the problems in Ape210k are more complex and difficult, and the model is more likely to miss them). Among them, the numerical gaps of KnBERT-TD in arithmetic problems and equation problems, Ape-clean and Ape-unsolvable, are not particularly large, and are smaller than the gaps of other baselines. This indicates that although KnBERT-TD's ability to handle difficult problems is still not at the level of simple problems, it is higher than that of most current solution models. It is also sufficient to prove that KnBERT-TD has stronger reasoning ability and superiority in mathematics.

Error analysis

This paper evaluates the predictive performance of the model in specific tasks through error analysis and selects the F1 score as the main error indicator. The range of the F1 score is between 0 and 1, and the closer the value is to 1, the better the overall performance of the model in terms of accuracy and recall.

In the experiment, we combined the recall and accuracy of the model to calculate an F1 score of 0.84. By comparing our model with other models such as GPT-4 and BERT-CL, the results showed that our F1 score was closer to 1. This result indicates that our model performs well in predicting positive samples, balancing accuracy and recall reasonably. Our model's advantage in positive sample prediction ability better meets practical business needs, enhancing the credibility and practicality of the model in real-world applications.

Ablation experiment

In this section, the influence of pre-training, masking strategies, and pre-training tasks on accuracy is comprehensively explored through ablation experiments. The effect of the judgment mechanism on training efficiency is also evaluated. Because of the excellent performance of the solver on Chinese MWPs, these experiments are carried out mainly on the Chinese dataset Math23K. Only the comparative data of the English dataset are included in "Impact of Pre-Training".

Impact of pre-training

In "Pre-training", the main function of pre-training in the solution is discussed. As shown in Table 4, the pre-trained model is more than 10% more accurate than the untrained model on Math23K, while it is 7% more accurate on MathQA. This experimental result proves the importance of pre-training for text tasks.

Impact of masking strategies

As shown in Table 5, three masking strategies are added to the model in turn on the basis of pre-training. It is observed that both phrase masking and entity masking have a positive impact on performance. It is worth mentioning that the role of entity masking is particularly significant. The entity masking is 2.8% more accurate than the previous one, while the phrase masking is only 1.6% more accurate. This is because the implicit

Items	Word-related tasks		Structure-related tasks		Semantic-related tasks
	+CP	+KP	+SR	+SPJ	+DRP
Accuracy	82.5%	83.0%	83.7%	84.1%	85.7%

Table 6. Effect of pre-training tasks on accuracy.

Items	Window size			Step size			Pooling type	
	2*2	4*4	6*6	1	2	3	Mean	Maximum
Accuracy	83.3%	85.7%	85.1%	84.3%	85.7%	84.7%	85.7%	81.5%

Table 7. Effect of *Mean Pooling*.

Item	Se2Seq	Seq2Tree
Accuracy	79.7%	85.7%

Table 8. The effect of the tree decoder.

Threshold	0	0.2	0.4	0.6	0.7	0.8	0.9	1.0
Accuracy	15.0%	23.1%	40.3%	72.5%	85.7%	85.7%	85.7%	85.7%

Table 9. Confidence thresholds comparison.

relationship between entities is crucial to the model's understanding of the problem text. This information greatly improves the model's understanding and problem-solving ability.

Impact of pre-training tasks
As shown in Table 6, among the three types of tasks, the semantic-related tasks have the most significant impact on the model. The improvement in accuracy brought by structure-related tasks is less than 1%, while semantic-related tasks directly increase it by 1.6-2%. This may be because the prediction of discourse relations can effectively capture semantic connections between contexts, thus helping the model obtain an in-depth understanding of the text content. In addition, with the gradual learning of each pre-training task, the accuracy metric continuously improves, which proves the semantic effectiveness of the pre-training tasks in improving the model's understanding ability.

Impact of mean pooling
This section examines the setting of different pooling hyperparameters, including pooling window size, pooling step size, and pooling type (maximum versus mean pooling), and explores the impact of each factor on model performance. The experimental results show that the size of the pooled window has a significant impact on information aggregation. Among them, the accuracy achieved under the 4*4 pooling window is 0.6 to 2.4% higher than that of the other two sizes, and the pooling step size 2 is about 1% higher than steps 1 and 3. The most important aspect regarding the selection of pooling modes is that it can be seen that the accuracy of mean pooling is 4.2% higher than that of maximum pooling, which is sufficient to prove that mean pooling is more suitable for this solution task. As shown in Table 7.

Impact of tree-decoder
As shown in Table 8, the accuracy brought by the tree decoder is 6% higher than that of traditional sequence decoding. Because it can effectively capture the hierarchy of mathematical equations, the model can better understand the relationship between the different components of the equation. The architecture of the tree decoder can avoid generating invalid expressions and duplicate equations. In addition, with the gradual improvement of the tree structure during training, the ability of the model to generate accurate equations is continuously improved, proving the advantages of using the tree-based method to construct mathematical problem-solving tasks.

Impact of confidence thresholds
Experiments are set to observe the effects of different thresholds on the model performance to better demonstrate the effectiveness of confidence estimation. As can be seen from Table 9, the smaller the threshold value, the lower the solving accuracy of the model; When the threshold exceeds 0.7, the learning ability of the model has reached the limit, and the improvement of the solution accuracy is not large. Therefore, any value from 0.7 onwards can

Item	w/o JM	w JM
Accuracy	17.5 h	11.3 h

Table 10. The effect of the judgment mechanism on training efficiency.

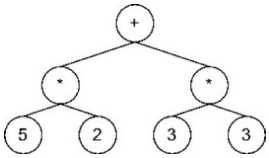

Fig. 6. Binary expression tree.

be used as a definite threshold for the final model. In the specific task, according to the detection error size of different thresholds, we choose the smallest error as the final threshold. Based on our results, 0.7 was chosen as the final threshold.

Impact of judgment mechanism
As shown in Table 10, the judgment mechanism (JM) effectively shortens the training time of the model. After adding the judgment mechanism, the model training time was reduced by six hours, effectively saving time and computational data. The reason for this is that the model selects a trustworthy inference after it predicts the expression, and then it computes the expression. This avoids wasting computing resources on errors or uncertain problems, thus effectively reducing unnecessary computation and significantly improving training efficiency.

Case analysis

While deep learning models are excellent at handling a wide range of language tasks, their internal workings are often viewed as a black-box process that is difficult for humans to intuitively understand, which limits the interpretability of the models. Therefore, a simple application problem example is presented to show the flow of the method.

Problem input and preprocessing
Input: Xiao Ming bought 5 apples, each apple 2 yuan, he also bought 3 oranges, each orange 3 yuan, Xiao Ming spent a total of how much money?

Preprocessing: Converting text input into a machine-processable format, including word segmentation, part-of-speech tagging, named entity recognition, and so on. For example, "apple" and "orange" are identified as item nouns, and "5" and "2 yuan" are described as quantity and price.

Semantic analysis
Deep semantic understanding: Using the language model KnBERT to understand semantic relationships in sentences, identify key information (such as items, quantities, unit prices) and their relationships.

Feature extraction: Transform text information into a vector representation to provide a basis for subsequent processing.

Quantitative relation extraction and inference
Direct quantity relationship: The number of apples is 5, and the unit price is 2 yuan; the quantity of oranges is 3 and the unit price is 3 yuan.

Implicit quantity relation: Total price needs to be calculated, that is, total price = total price of apples + total price of oranges.

Expression tree generation: The resulting expression tree is shown in Fig. 6.

Traversing the tree: The expression is "Total price = 5*2 + 3*3".

Expression evaluation and solution
Judge the expression: The confidence value $c>$ the threshold δ, indicating that the expression is trustworthy, and subsequent calculation is performed.

Answer generation: Output the final answer "Xiao Ming spent a total of 19 yuan".

Conclusions

In this study, we proposed a machine solution for MWPs based on semantic understanding enhancement. A special knowledge-enhanced pre-training language model named KnBERT was designed. A range of knowledge masking strategies and pre-training tasks ensures that KnBERT learns background knowledge and more semantic information, and they enhance the understanding of problem texts. Mean pooling was introduced to collect the semantic information in the region and assist KnBERT in overcoming the semantic understanding bias of MWPs. In addition, we designed a novel solver named KnBERT-TD. The pre-trained model acts as a semantic encoder, and a binary tree acts as a decoder, which improves the solving accuracy and guarantees the uniqueness

and validity of the equation. Finally, a judgment mechanism was proposed. This mechanism improves the training efficiency while ensuring the accuracy of the solution. The experimental results show that KnBERT-TD achieved the highest accuracy on Chinese and English datasets when compared with other baselines, and ablation experiments also demonstrated the effectiveness of the optimization measures and judgment mechanism. In the future, the machine solution could also develop intelligent teaching auxiliary functions such as homework intelligent correction and an automatic question-answering system, contributing to the intelligent development of the education field. However, although this method has demonstrated great performance in most experiments, it still has certain limitations. Firstly, the selection of the threshold depends on the specific characteristics of the dataset. Therefore, on different tasks and datasets, additional adjustments and verifications may be required. Secondly, the introduction of confidence branches may increase the complexity of the model. For systems with limited computing resources, this method may lead to certain performance bottlenecks. Finally, the "black box" feature of neural network reasoning makes the reasoning process of the model less intuitive, which is only proposed in this paper. The solution theorem of binary trees is far from sufficient, which is a constraint for understanding the reasoning mechanism of the model and further optimizing the model. In response to these challenges, our future research should also continue to explore.

Discussion

Although this paper has proposed solutions in terms of semantic understanding and precise solutions, there are still other unsolved problems in the field of MWP. MWPs often involve knowledge in multiple fields, such as physics, chemistry, biology, and so on. Future research on MWPs can explore how to achieve cross-domain knowledge fusion so that machines can use knowledge from multiple fields to solve word problems. This may require building a richer knowledge base and studying how to effectively connect and integrate knowledge from different fields. Moreover, in practical applications such as education, scientific research, industry, etc., there are various needs for machines to solve MWPs. In the future, we can pay attention to these actual demands and adjust and optimize the problem-solving strategies and methods of machines according to specific scenarios. For example, in the field of education, we can study how to provide individualized problem-solving guidance according to students' learning characteristics and needs. In scientific research, we can explore how to use machines to solve word problems to assist researchers in data processing and analysis.

Data availability

The data that support the findings of this study are available on request from the corresponding author, [M Y], upon reasonable request.

Received: 28 April 2024; Accepted: 12 September 2025
Published online: 21 October 2025

References

1. Minaee, S. et al. Deep learning–based text classification: a comprehensive review. *ACM Comput. Surv. (CSUR).* **54** (3), 1–40 (2021).
2. Khashabi, D. et al. Unifiedqa: Crossing format boundaries with a single qa system. https://arXiv.org/abs/2005.00700. (2020).
3. Wang, Y., Liu & Shi, S. Deep neural solver for math word problems. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. (2017).
4. Wang, L. et al. Template-based math word problem solvers with recursive neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (01). (2019).
5. Devlin, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. https://arXiv.org/abs/1810.04805. (2018).
6. Radford, A. et al. Improving language understanding by generative pre-training. (2018).
7. Yang, Z. et al. Xlnet: generalized autoregressive pretraining for Language Understanding. *Advances Neural Inform. Process. Syst.* **32** (2019).
8. Zhang, Y. et al. Dialogpt: Large-scale generative pre-training for conversational response generation. https://arXiv.org/abs/1911.00536. (2019).
9. Li, Z. et al. Seeking patterns, not just memorizing procedures: contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: ACL 2022* 2486–2496. (Association for Computational Linguistics, 2022).
10. Liang, Z. et al. Mwp-bert: Numeracy-augmented pre-training for math word problem solving. https://arXiv.org/abs/2107.13435. (2021).
11. Sun, Y. et al. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*. **34** (05). (2020).
12. Z. & Sun, S. A goal-driven tree-structured neural model for math word problems. *Ijcai* (2019).
13. Zhang, J. et al. Graph-to-tree learning for solving math word problems. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. (2020).
14. Chen, Y. *Convolutional neural network for sentence classification*. MS thesis. University of Waterloo, (2015).
15. Gehring, J. et al. Convolutional sequence to sequence learning. *International conference on machine learning*. (PMLR, 2017).
16. Nakano, R. et al. Webgpt: Browser-assisted question-answering with human feedback. https://arxiv.org/abs/2112.09332 (2021).
17. Wei, J. et al. Chain-of-thought prompting elicits reasoning in large Language models. *Adv. Neural. Inf. Process. Syst.* **35**, 24824–24837 (2022).
18. Turing, A. M. *Computing Machinery and Intelligence* (Springer Netherlands, 2009).
19. Bobrow, D. Natural language input for a computer problem solving system. (1964).
20. Wu Wenjun. Elementary geometric decision problem and mechanized proof. *Science* **6** (1977).
21. Zhang Jingzhong, Y., Lu & Xiaorong, H. WE complete method for machine proof of geometric theorems. *Syst. Sci. Math.* **15** (3), 200–207 (1995).
22. Wu Wenjun. Basic principles of machine proof of elementary geometric theorems. *Syst. Sci. Math.* **3** (1984).
23. Xu Liang. Research on automatic addition of auxiliary lines (points) and its application in solid geometry. MS thesis. University of Electronic Science and Technology of China. (2017).
24. Gan Wenbin. Research on automatic solution of plane geometry problems. *Cent. China Normal Univ.* (2018).
25. Bakman, Y. Robust understanding of word problems with extraneous information. https://arXiv.org/abs/0701393. (2007).

26. Kushman, N. et al. Learning to automatically solve algebra word problems. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. (2014).
27. Shi, S. et al. Automatically solving number word problems by semantic parsing and reasoning. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. (2015).
28. Liang, C. C. et al. A tag-based English math word problem solver with understanding, reasoning and explanation. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. (2016).
29. Wang, L. et al. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence* **32** (1). (2018).
30. Kim, B. et al. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020).
31. Lan, Z. et al. Albert: A lite bert for self-supervised learning of language representations. https://arXiv.org/abs/1909.11942. (2019).
32. Tan, M. et al. Investigating math word problems using pretrained multilingual language models. https://arXiv.org/abs/2105.08928. (2021).
33. Huang, S. et al. Recall and learn: A memory-augmented solver for math word problems. https://arXiv.org/abs/2109.13112. (2021).
34. Sileo, D. et al. Mining discourse markers for unsupervised sentence representation learning. https://arXiv.org/abs/1903.11850. (2019).
35. Yan, M. *International Conference on Intelligent Education and Intelligent Research (IEIR)* (IEEE, 2023).
36. DeVries, T. & Graham, W. Taylor. Learning confidence for out-of-distribution detection in neural networks. https://arXiv.org/abs/1802.04865. (2018).
37. Zhao, W. et al. Ape210k: A large-scale and template-rich dataset of math word problems. https://arXiv.org/abs/2009.11506. (2020).
38. Amini, A. et al. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. https://arXiv.org/abs/1905.13319. (2019).
39. Wang, L. et al. Translating a math word problem to an expression tree. https://arXiv.org/abs/1811.05632. (2018).
40. Zhang, J. et al. Teacher-student networks with multiple decoders for solving math word problem. *IJCAI*, (2020).
41. Lan, Y. et al. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. *Proceedings of the AAAI Conference on Artificial Intelligence* **36** (11). (2022).
42. Achiam, J. et al. Gpt-4 technical report. https://arXiv.org/abs/2303.08774. (2023).
43. Dixit, P. and Tim Oates. Sbi-rag: Enhancing math word problem solving for students through schema-based instruction and retrieval-augmented generation. https://arXiv.org/abs/2410.13293. (2024).
44. Kai, D., Zhenguo, M. & **aoran, Y. Logic contrastive reasoning with lightweight large language model for math word problems. https://arXiv.org/abs/2409.00131. (2024).

Acknowledgements

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers and editors, which improved the presentation of this research.

Author contributions

All authors contributed to the study's conception and design. Y.W. and P.J. designed research methods and experiments and provides experimental equipment, technology and financial support. M.Y. and Y.W.were responsible for the planning, design and implementation of the entire study, writing and revising the paper. Y.Y. and Y.L. participated in the study design, and method selection, and assisted in data collection, processing, and analysis. All authors read and approved the final manuscript.

Funding

This work is supported by the National Natural Science Foundation of China (No. 62107014), the Youth Talent Support Project of Henan Province (2023HYTP046), and the Philosophy and Social Sciences Project of Henan Province (2024CKS026).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to M.Y.