Applied Intelligence (2023) 53:14744–14758
https://doi.org/10.1007/s10489-022-04253-1

Goal selection and feedback for solving math word problems

Daijun He[1] · **Jing Xiao**[1]

Accepted: 9 October 2022 / Published online: 2 November 2022

Abstract
Solving Math Word Problems (MWPs) automatically is a challenging task for AI-tutoring in online education. Most of the existing State-Of-The-Art (SOTA) neural models for solving MWPs use Goal-driven Tree-structured Solver (GTS) as their decoders. However, owing to the defects of the tree-structured recurrent neural networks, GTS can not obtain the information of all generated nodes in each decoding time step. Therefore, the performance for long math expressions is not satisfactory enough. To address such limitations, we propose a Goal Selection and Feedback (GSF) decoding module. In each time step of GSF, we firstly feed the latest result back to all goal vectors through goal feedback operation, and then the goal selection operation based on attention mechanism is designed for generate the new goal vector. Not only can the decoder collect the historical information from all generated nodes through goal selection operation, but also these generated nodes are always updated timely by goal feedback operation. In addition, a Multilayer Fusion Network (MFN) is proposed to provide a better representation of each hidden state during decoding. Combining the ELECTRA language model with our novel decoder, experiments on the Math23k, Ape-clean, and MAWPS datasets show that our model outperforms the SOTA baselines, especially on the MWPs of complex samples with long math expressions. The ablation study and case study further verify that our model can better solve the samples with long expressions, and the proposed components are indeed able to help enhance the performance of the model.

Keywords NLP · MWPs · Attention · Decoding · Goal selection and feedback

1 Introduction

In recent years, the development of online educational applications has been greatly accelerated, especially after the covid-19 outbreak. Automatically scoring the students' answers in online education applications can ease the burden on teachers of marking a large number of repetitive assignments. However, in Math Word Problems (MWPs), the performance of automatically solving is far from perfect and cannot yet be applied in practice. MWPs is a mathematical problem described in natural language. At the end of the MWPs' text, the answer to the mathematical problem is required. MWP solving is very helpful to cultivate students' mathematical ability of problem analysis and calculation. Table 1 is an example of a typical MWP where the reader is required to answer the number of motorcycles in the parking lot. In automatic MWPs solving task, a machine must deduce an answer to a given mathematical problem by acquiring the implied numeric information in the problem.

In the 1960s, some researchers began to adopt Artificial Intelligence (AI) methods to solve MWPs. Recently, Wang et al. used a modified Seq2Seq model to build the MWPs Solver [1] and more and more researchers proposed a variety of approaches based on deep learning for solving MWPs [2–6]. Deep learning-based methods turn MWP solving into a natural language generation task, where the generated text is not "Natural", but consists of mathematical symbols. The crucial advantage of the Deep Learning-based models is that they eliminate the need for hand-crafted features.

Nowadays, most SOTA models use the decoder of Goal-Driven Tree-structured MWPs Solver (GTS) proposed by Xie and Sun [4] for model decoding and token prediction [5–7]. The goal-driven mechanism in human problem

✉ Jing Xiao
xiaojing@scnu.edu.cn

Daijun He
2020022959@m.scnu.edu.cn

[1] School of Computer Science, South China Normal University, No. 55 Zhongshan West Road, Guangzhou, 510631, Guangdong, China

Table 1 A math word problem

Problem: In a parking lot, there are 48 cars and motorcycles. If threre is a total of 172 wheels altogether, where each car has 4 wheels and each motorcycle has 3 wheels, what is the number of motorcycles?
Expression: $(48 \times 4 - 172) \div (4 - 3)$ Solution: 20

solving is proposed by GTS. When humans read the text of a math word problem, they first figure out which target quantity is to be derived as the goal, and then pay attention to the relevant information of the problem which can help to realize the goal [4]. The decoding process of GTS' decoder follows the generation process of the binary tree and each node corresponds to one goal to be solved.

Figure 1 shows a typical example of a mistake made by the GTS when solving the math problem in Table 1. The token "3" predicted by goal vector $\mathbf{q}_7$ shows that root goal vector $\mathbf{q}_1$ wants to solve the problem by template "[Total number of motorcycles' wheels] ÷ [Number of wheels each motorcycle has]". Then, goal vector $\mathbf{q}_2$ uses another approach to solve this problem, rather than following $\mathbf{q}_1$ to calculate [Total number of motorcycles' wheels]. However, goal vector $\mathbf{q}_7$ only obtains two kinds of historical information through two recurrent neural networks (RNNs): ① information of each ancestor node's goal vector (e.g. $\mathbf{q}_1$) through top-down goal decomposition (blue arrow); ② information of embeddings of all generated tokens through bottom-up subtree embedding (green arrows). That is to say, $\mathbf{q}_7$ can not get the information about the goal vectors of all generated nodes (i.e. $\mathbf{q}_2$, $\mathbf{q}_3$, $\mathbf{q}_4$, $\mathbf{q}_5$, $\mathbf{q}_6$) except the ancestor node $\mathbf{q}_1$. In the meantime, the embeddings of generated tokens can not reflect the changes in the solution strategy. Therefore, goal vector $\mathbf{q}_7$ is not notified of solution strategy changes, and still follows $\mathbf{q}_1$ to find [Number of wheels each motorcycle has]. This defect makes it difficult for GTS to handle the samples with complex math expressions.

To address this issue, we design a new goal-driven decoding approach which is called **Goal Selection and Feedback** (GSF). Figure 2 shows the process of generating

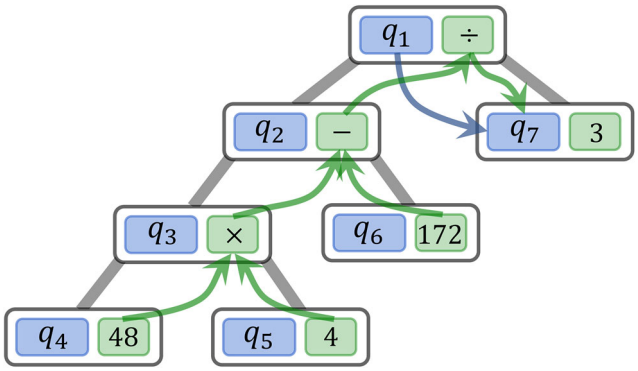


Fig. 1 The process of GTS' node generation

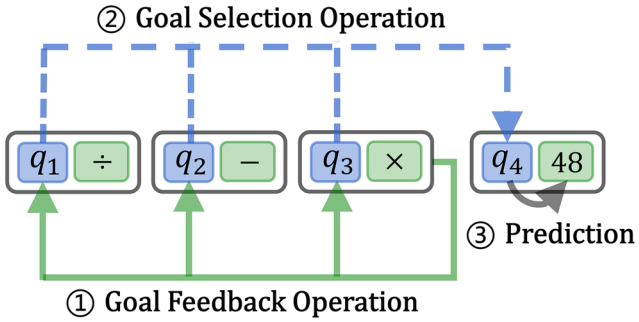


Fig. 2 The process of Goal Selection and Feedback

the 4th node through GSF. Firstly, the Goal Feedback Operation updates the information of the 3th node to each goal vector (green arrows), allowing them to adjust themselves at each time step to provide more accurate information for the current decoding step. Then, the Goal Selection Operation selects the information from the updated previous goal vectors directly and generates the new goal vector $\mathbf{q}_4$ by attention mechanism (blue dotted arrow). Finally, token "48" is predicted according to $\mathbf{q}_4$ and context from encoded problem text (gray arrow). This approach allows the decoder to capture the most related information from all generated nodes in each time step. Because all generated nodes would be updated by the goal feedback operation according to the latest result and be provided to the goal selection operation. Moreover, we design **Multilayer Fusion Network** (MFN) to enhance the information fusion capacity instead of using a multilayer perceptron, which leads to a better representation for each hidden state. At last, we use ELECTRA language model [8] as the encoder. The proposed model is evaluated on the Math23k dataset, Ape-clean dataset, and MAWPS dataset. Experimental results show that our model can achieve better performance against strong baselines.

The contributions of this paper are summarized here:

- We propose the Goal Selection and Feedback (GSF) decoding approach, in which the goal feedback operation feds the latest result back to each generated goal, and the goal selection operation selects the updated past goals for decoding.
- We design the Multilayer Fusion Network (MFN) to model the hidden states instead of using a multilayer perceptron.
- The experimental results show our model outperforms several SOTA systems on the dataset Math23k.

2 Related work

In the first stage of the MWPs solving, from 1960 to 2010, systems such as STUDENT [9], DEDUCOM [10],

WORDPRO [11] and ROBUST [12], manually designed rules and schemas for pattern matchings. Yun et al. also use schema for multi-step math problem solving, but the implementation details are not explicitly revealed [13]. These methods are complicated and difficult to reproduce.

After 2010, some researchers started to employ machine learning methods in solving MWPs. The MWPs solvers designed by them predicted and filled the predefined expression templates through traditional machine learning methods [14, 15]. Meanwhile, another way to solve MWPs is to use semantic parsing, which mapped the problem text to structured logic forms and inferred the answer through the predefined logic rules [16, 17]. Obviously, all approaches above required tremendous human efforts in feature engineering and annotation, which result in poor generality.

In recent years, deep learning has made great progress in various domains, such as machine translation [18–20], object detection [21–23], text classification [24–26], and dialogue system [27–29]. It is not surprising to notice that MWPs can be better solved with DL-based methods. The first DL-based MWP solver was an improved Seq2Seq model, which has been widely applied to translation tasks and question-answer tasks [1]. In order to improve the generality of DL-based models for solving MWPs, researchers proposed Significant Number Identification (SNI) and Equation Normalization (EN) [1, 2]. Due to the fact that mathematical expressions can be converted into binary trees, some researchers discarded the linear decoder in Seq2Seq and instead designed several brand new tree-structured recursive neural networks for model decoding, such as abstract syntax tree decoder (AST-Dec) [30] and GTS [4].

With the prevailing application of GTS, some researchers focused on improving the model's understanding of the problem text and used GTS as the decoder of their models. Zhang et al. proposed a novel graph-based encoder to learn the quantity-related features for enhancing problem understanding [5]. Imitating human reading habits, Lin et al. proposed a hierarchical word-clause-problem encoder and applied a hierarchical attention mechanism to enhance the problem semantics with context from different levels, and a pointer-generator network to guide the model to copy existing information and infer extra knowledge in decoding [6].

In addition to designing DL-based MWPs solvers, some researchers look for other ways to enhance the accuracy of the model. Shen et al. devise a new ranking task for MWP and proposed Generate & Rank, a multi-task framework based on a generative pre-trained language model, the model learns from its own mistakes and is able to distinguish between correct and incorrect expressions [31]. Instead of using GTS, Lee et al. proposed a TM-generation model that uses the decoder of Transformer to predict the math expression templates, and then fills the missing operators in the predicted templates by the operator identification layer that they designed [32].

3 Problem definition and data processing

The formal definition of a Math Word Problem sample is (P, E) and the data processing is shown in Fig. 3, where:

- The problem text P is a sequence of n word tokens or numeric values: $P = \{p_1, p_2, ..., p_n\}$, where p_i is either a word token or a numeric value. According to SNI [1], the ith numeric value p_k appearing in P is denoted as N_i. We replace p_k with the token "NUM" in the original sequence P. At the end, the start token "SOS" and the ending token "EOS" are added to the beginning and the end of the sequence respectively. Finally we get a sequence P' of $m = n + 2$ tokens.
- The math expression E derived from problem text P is used to calculate the unknown quantity required by MWP. Expression E is defined as a sequence containing math operators from V_{op}, numeric constants from V_{con}, and denoted numeric values in n_P from the problem text P. Therefore, the output vocabulary of the our MWP solver is $V_{dec} = V_{op} \cup V_{con} \cup n_P$. We replace all numeric values from the original expression with the corresponding number token $N_i \in n_P$ and convert the expression into the corresponding prefix form. In the end, the start token "SOS" and the ending token "EOS" are added to the beginning and the end of the expression respectively.

Given the problem text P, our model computes the output sequence $\hat{E}$ according to P.

4 Model description

4.1 Overview

To address the shortcomings of GTS, we propose a Goal-Driven MWP Solver with Goal Selection and Feedback. We follow the idea of the goal-driven mechanism, which means that our model would generate a new goal vector at each time step to guide the model to find the solution. Figure 4 shows the overview structure of our model. First, we input the processed text into the ELECTRA language model for encoding. Second, through the main goal initialization, we get the main goal vector $\mathbf{q}_0$, which represents the problem in the text that we need to solve. At the same time, we assume that the initial token "SOS" is generated by $\mathbf{q}_0$, which is the starting point of the expression generating process.

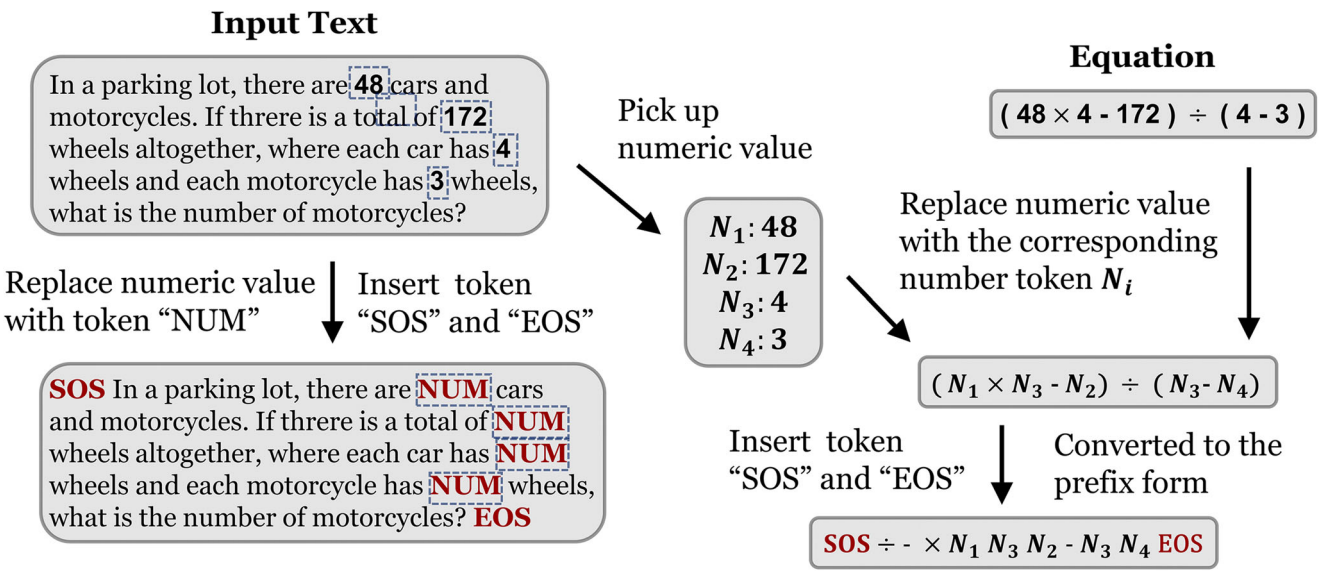


Fig. 3 Data processing

As we can see from the Fig. 4, the computation of our model has two directions: Goal Feedback Operation in the vertical direction (green half arrow) and Goal Selection Operation in the parallel direction (blue half arrow). In the vertical direction, each generated goal vector adjusts itself according to the last generated result by the Goal Feedback Operation (green arrows and orange arrows in Fig. 4). After that, each new goal vector is generated by the Goal Selection Operation (black dotted boxes and blue dotted arrows in Fig. 4). The Goal Selection Operation is based on the attention mechanism, which means that every goal vector in the black dot boxes has the opportunity to affect the generation of the latest goal vector. On the other hand, we treat the latest output token as the partial solution to the problem, and each goal would be resolved to some extent when a solution comes out. Therefore, we believe that when the latest result is generated, each goal vector should be adjusted itself to provide more accurate information for the generation of the next goal vector.

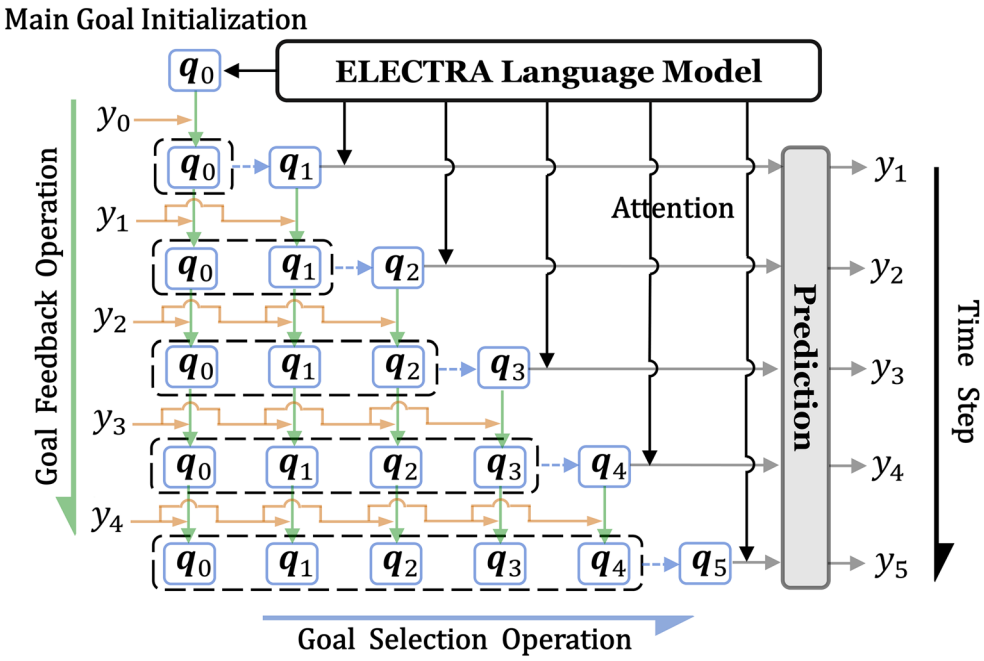


Fig. 4 Overview of our proposed model

When the latest goal vector is generated, our model summarizes relevant information into the context vector from the encoded text through the attention mechanism (black arrows in Fig. 4). Using the goal vector and the context vector, we score every token $y \in V_{dec}$ and the token with the highest score is selected as the output of decoding (gray arrows and gray box in Fig. 4). Then the above steps are repeated until the ending token "EOS" is decoded.

4.2 Electra language model

In order to enhance the model's understanding of the problem text, ELECTRA language model [8] is chosen as the encoder of our model, instead of encoding the problem text sequence simply through the RNN, such as LSTM and GRU. The authors of ELECTRA proposed a more sample-efficient pre-training task called replaced token detection and trained a discriminative model to predict whether each token was replaced or not. Experiments show that the contextual representations learned by the ELECTRA language model substantially outperform the ones learned by BERT given the same model settings [8].

After tokenization, the input problem text is converted to $X = \{x_1, x_2, ..., x_m\}$, and each token x_k indicates the position of the corresponding word token p_k in the input vocabulary. Then, the sequence X is input to the ELECTRA language model.

$$H = \text{ELECTRA}(X) \tag{1}$$

where ELECTRA$(\cdot)$ denotes the function of ELECTRA language model. The ELECTRA language model accepts a token sequence X as input, and outputs the encoded sequence $H = \{\mathbf{h}'_1, \mathbf{h}'_2, ..., \mathbf{h}'_m\}$ through a series of calculations, such as multi-head attention, self-attention, and position-wise feed-forward networks. Then, we calculate $\mathbf{h}_i$ as follows, and treat it as the contextual representation of the token $p_i \in P'$.

$$\mathbf{h}_i = \text{LN}(\mathbf{W}_e \mathbf{h}'_i), \quad i = 1, 2, ..., m \tag{2}$$

where LN$(\cdot)$ denotes the layer normalization layer [33], $\mathbf{W}_e \in \mathbb{R}^{d \times d}$ is a trainable matrix and d is the dimensionality of the ELECTRA's output.

4.3 Dynamic token embedding

To make the prediction of the model more accurate, we use dynamic token embedding like GTS [4]. Diffrent from operator tokens and constant tokens, token $y = N_i \in N_P$ treats the contextual representation vector $\mathbf{h}_j$ as its embedding, where j is the index position of N_i in the problem text P'. Each token $y \in V_{op} \cup V_{con}$ has the same embedding in all problems.

$$\mathbf{e}(y) = \begin{cases} \mathbf{M}_s(y) & \text{if } y \in V_{op} \cup V_{con} \\ \mathbf{h}_{loc(y,P')} & \text{if } y \in N_P \end{cases} \tag{3}$$

where $\mathbf{e}(\cdot)$ denotes the embedding mapping function, $\mathbf{M}_s \in \mathbb{R}^{d \times |V_{op} \cup V_{con}|}$ is a trainable embedding matrix, and $loc(y, P')$ is the index position of y in P'. In this way, every token from the common part $V_{op} \cup V_{con}$ of all problems' target vocabulary has the same representations in all problems, while the representations of token $y = N_i \in N_P$ in each problem are different. Such embedding mapping is intuitive and the meaning of the same numeric value token N_i in every problem must be different.

4.4 Multilayer fusion network

To improve the information fusion capacity of our network, we propose the Multilayer Fusion Network (MFN) to generate several hidden states. Given the inputs $\mathbf{p}$ and $\mathbf{g}$, the n-layer MFN calculates the result through the following iterative steps:

$$\begin{aligned} \mathbf{o}_0 &= \text{ReLU}(\mathbf{W}_0[\mathbf{p}, \mathbf{g}]) \\ \mathbf{s}_{k+1} &= \text{ReLU}(\mathbf{W}_{s_{k+1}}[\mathbf{p}, \mathbf{g}]) \\ \mathbf{z}_{k+1} &= \text{ReLU} \quad (\mathbf{W}_{z_{k+1}}[\mathbf{p}, \text{LN}(\mathbf{o}_k), \mathbf{s}_{k+1}]) \\ \mathbf{o}_{k+1} &= \mathbf{o}_k - \quad \mathbf{s}_{k+1} * \mathbf{z}_{k+1}, \quad k = 0, 1, 2, ..., n-1 \\ & \quad \text{MFN}(\mathbf{p}, \mathbf{g}) = \text{LN}(\mathbf{o}_n) \end{aligned} \tag{4}$$

where $\mathbf{W}_0 \in \mathbb{R}^{d \times 2d}, \mathbf{W}_{s_1} \in \mathbb{R}^{d \times 2d}, ..., \mathbf{W}_{s_n} \in \mathbb{R}^{d \times 2d}, \mathbf{W}_{z_1} \in \mathbb{R}^{d \times 3d}, ..., \mathbf{W}_{z_n} \in \mathbb{R}^{d \times 3d}$ are trainable matrices, $[\cdot, \cdot, ..., \cdot]$ denotes concatenation, * denotes the hadamard product and ReLU$(\cdot)$ denotes the ReLU function. $\mathbf{o}_0$ is the initial vector of combining the information of $\mathbf{p}$ and $\mathbf{g}$. $\mathbf{s}_k$ is the kth *infromation supply* vector, and $\mathbf{z}_k$ is the kth *scale* gate of $\mathbf{s}_k$. The kth scaled supply vector $\mathbf{s}_k * \mathbf{z}_k$ decides what information should be added to $\mathbf{o}_k$. Through subtraction, $\mathbf{o}_k$ receives the information provided by $\mathbf{s}_k * \mathbf{z}_k$. The iterative multilayer structure is designed to allow MFN to provide deeper information fusion.

4.5 Goal selection and feedback

Main goal initialization To start the decoding process, we use token y_0 ="SOS" as the start token of decoding, and the corresponding initial goal vector $\mathbf{q}_0$ and the initial prediction vector $\mathbf{u}_0$ are calculated as follows:

$$\begin{aligned} \mathbf{q}_0 &= \mathbf{h}_1 \\ \mathbf{u}_0 &= \text{LN}(\text{ReLU}(\mathbf{W}_m \mathbf{q}_0)) \end{aligned} \tag{5}$$

where $\mathbf{h}_1$ is the contextual representations of the token "SOS" $\in P'$, and $\mathbf{W}_m \in \mathbb{R}^{d \times d}$ is a trainable matrix. The

initial goal vector $\mathbf{q}_0$ represents the main goal to be solved, which is posed by the problem text directly. With the initial token and the corresponding goal vector $\mathbf{q}_0$ and prediction vector $\mathbf{u}_0$, the decoder performs each step of decoding through the following iterative steps.

Goal feedback operation Given a main goal vector $\mathbf{q}_0$, the kth prediction vector $\mathbf{u}_k$ and the kth output token y_k, the decoder merges them through a single layer perceptron to get $\mathbf{r}_k$ as the kth solution vector. Then, the *update* gate g_i^k of the ith goal vector is calculated as follows:

$$\begin{aligned} \mathbf{r}_k &= \mathrm{LN}(\mathrm{ReLU}(\mathbf{W}_r[\mathbf{q}_0, \mathbf{u}_k, \mathbf{e}(y_k)])) \\ g_i^k &= \sigma(\mathbf{v}_g^\top \mathrm{ReLU}(\mathbf{W}_g[\mathbf{r}_k, \mathbf{q}_i])) \end{aligned} \tag{6}$$

where $\mathbf{v}_g \in \mathbb{R}^d$, $\mathbf{W}_r \in \mathbb{R}^{d\times 3d}$ and $\mathbf{W}_g \in \mathbb{R}^{d\times 2d}$ are trainable parameters, and $\sigma(\cdot)$ denotes the sigmoid function. Then, each goal vector adjust themselves according to the update gate g_i^k and the feedback vector $\mathbf{f}_i^k$ calculated by the trainable network MFN_f as follows (green arrows in Fig. 5)

$$\begin{aligned} \mathbf{f}_i^k &= \mathrm{MFN}_f(\mathbf{q}_i, \mathbf{r}_k) \\ \mathbf{q}_i &:= (1 - g_i^k) \cdot \mathbf{q}_i + g_i^k \cdot \mathbf{f}_i^k, \quad i = 0, 1, 2, ..., k \end{aligned} \tag{7}$$

Goal selection operation After the goal feedback operation, we calculates the selection weight a_i^k by kth solution vector $\mathbf{r}_k$ and each goal vector $\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_k$ (blue arrows and blue dotted arrows in Fig. 5):

$$\begin{aligned} \mathrm{S}_g(\mathbf{r}_k, \mathbf{q}_i) &= \mathbf{v}_s^\top \mathrm{ReLU}(\mathbf{W}_s[\mathbf{r}_k, \mathbf{q}_i]) \\ a_i^k &= \frac{\exp(\mathrm{S}_g(\mathbf{r}_k, \mathbf{q}_i))}{\sum_s \exp(\mathrm{S}_g(\mathbf{r}_k, \mathbf{q}_s))} \end{aligned} \tag{8}$$

where $\mathbf{v}_s \in \mathbb{R}^d$ and $\mathbf{W}_s \in \mathbb{R}^{d\times 2d}$ are trainable parameters. Then the $(k+1)$th goal vector $\mathbf{q}_{k+1}$ is calculated as follows:

$$\begin{aligned} \tilde{\mathbf{q}}_i^k &= \mathrm{MFN}_s(\mathbf{r}_k, \mathbf{q}_i) \\ \mathbf{q}_{k+1} &= \sum_{i=0}^{k+1} a_i^k \tilde{\mathbf{q}}_i^k \end{aligned} \tag{9}$$

where MFN_s is a trainable network, $\tilde{\mathbf{q}}_i^k$ is the sub-goal vector derived by the solution vector $\mathbf{r}_k$ according to the ith goal vector $\mathbf{q}_i$, and weighted summation is performed to obtain the new goal vector $\mathbf{q}_{k+1}$.

Token prediction Given the goal vector $\mathbf{q}_{k+1}$, in order to accurately predict the output token, we compute the context vector $\mathbf{c}_{k+1}$ as follows to summerize the relevant information of the problem through the attention mechanism (black dotted arrow in Fig. 5):

$$\begin{aligned} \mathrm{S}_c(\mathbf{q}_{k+1}, \mathbf{h}_i) &= \mathbf{v}_c^\top \mathrm{ReLU}(\mathbf{W}_c[\mathbf{q}_{k+1}, \mathbf{h}_i]) \\ w_i &= \frac{\exp(\mathrm{S}_c(\mathbf{q}_{k+1}, \mathbf{h}_i))}{\sum_s \exp(\mathrm{S}_c(\mathbf{q}_{k+1}, \mathbf{h}_s))} \\ \mathbf{c}_{k+1} &= \sum_{i=1}^{m} w_i \mathbf{h}_i \end{aligned} \tag{10}$$

where $\mathbf{v}_c \in \mathbb{R}^d$ and $\mathbf{W}_c \in \mathbb{R}^{d\times 2d}$ are trainable parameters. Next, the prediction vector $\mathbf{u}_{k+1}$ is computed by combining the goal vector $\mathbf{q}_{k+1}$ with context vector $\mathbf{c}_{k+1}$, and then probability $\mathrm{prob}(y_i|\mathbf{u}_{k+1})$ of each token $y_i \in V_{dec}$ is calculated as (black arrows in Fig. 5):

$$\mathbf{u}_{k+1} = \mathrm{MFN}_c(\mathbf{q}_{k+1}, \mathbf{c}_{k+1}) \tag{11}$$

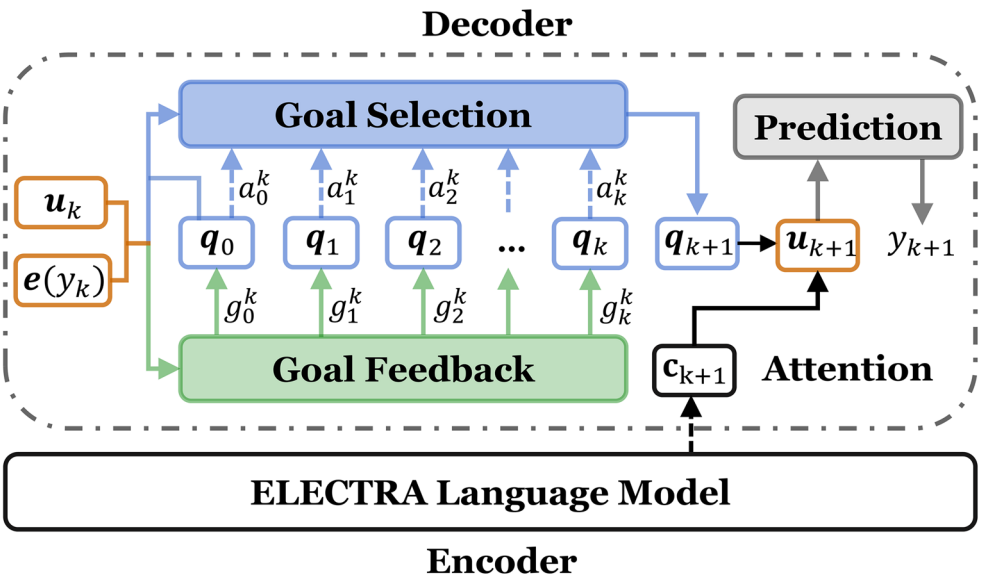


Fig. 5 Process of goal selection and feedback decoding approach

$$\begin{aligned} \mathrm{s}(y_i, \mathbf{u}_{k+1}) &= \mathbf{v}_p^\top \mathrm{ReLU}(\mathbf{W}_p[\mathbf{u}_{k+1}, \mathbf{e}(y_i)]) \\ \mathrm{prob}(y_i|\mathbf{u}_{k+1}) &= \frac{\exp(\mathrm{s}(y_i, \mathbf{u}_{k+1}))}{\sum_s \exp(\mathrm{s}(y_s, \mathbf{u}_{k+1}))} \end{aligned} \tag{12}$$

where MFN_c is a trainable network, $\mathbf{e}(y_i)$ is the token embedding of y_i calculated by (3), $\mathbf{v}_p \in \mathbb{R}^d$ is a trainable vector, and $\mathbf{W}_p \in \mathbb{R}^{d\times 2d}$ is a trainable matrix. Finally, token $\hat{y}_{k+1}$ with the highest probability is selected as the output token of the current decoding step (gray arrows in Fig. 5):

$$\hat{y}_{k+1} = \arg\max_{y\in V_{dec}} \mathrm{prob}(y|\mathbf{u}_{k+1}) \tag{13}$$

when the token $\hat{y}_{k+1}$ is "EOS", the decoder completes the decoding of the problem, otherwise, it continues with the next decoding step.

4.6 Model training

Formally, for each (P^i, E^i) in the training dataset $\mathbb{D} = \{(P^i, E^i)|1 \le i \le N\}$, P^i is the problem text sequence, and E^i is the math expression sequence corresponding to the problem P^i. The loss function $Loss_i$ is defined as the sum of the negative log-likelihood of probabilities for predicting tth token $y_t \in E^i$. The total loss function is calculated as follows:

$$\begin{aligned} Loss_i &= -\sum_{t=1}^{T} log\mathrm{prob}(y_t|\mathbf{u}_t) \\ Loss_{\mathrm{total}} &= \sum_{i=1}^{N} Loss_i \end{aligned} \tag{14}$$

where $\mathbf{u}_t$ is the prediction vector of the t-th node; T is the number of tokens in E^i, and $\mathrm{prob}(\cdot|\cdot)$ is computed by (12). The training goal of the model is to minimize $Loss_{\mathrm{total}}$.

5 Experiment

In this section, we compare our model with several SOTA baselines, then perform ablation experiments on each module, and finally end up with analyzing the experimental results.

5.1 Dataset and baselines

Dataset We conduct the experiments on the following datasets:

- **Math23k**[1]: The dataset Math23k [1] is a commonly-used large-scale Chinese MWPs dataset, containing 22,161 training problems and 1,000 testing problems with solution expressions and answers. Each math word problem can be solved by one linear algebra expression.
- **Ape-clean**[2]: The dataset Ape-clean [34] is the cleaned version of the Chinese MWPs dataset Ape210k [35]. After cleaning, Ape-clean contains 102,596 training problems and 2,422 testing problems. Each math word problem can be solved by one linear algebra expression.
- **MAWPS**[3]: The dataset MAWPS [36] contains English math word problems with one or more unknown variables. We select 2,353 problems with only one unknown variable and perform five-fold cross validation on it.

Baselines We compare the following methods on datasets Math23k and Ape-clean:

- **MathEN** [2]: The ensemble model selects the result according to the models' generation probability among BiLSTM, ConvS2S, and Transformer with equation normalization (EN).
- **GroupAtt** [37]: The Seq2Seq model with the group attention mechanism to extract global features, quantity-related features, quantity-pair features, and question-related features in MWPs respectively.
- **AST-Dec** [30]: This MWP solver uses LSTM for encoding and generates the abstract syntax tree of the equation in a top-down manner when decoding.
- **GTS** [4]: The Goal-Driven Tree-structured MWP Solver, using GRU for encoding.
- **Graph2Tree** [5]: The MWP Solver with graph-based encoder and GTS decoder.
- **SAUSolver** [38]: The semantically-aligned universal tree-structured solver based on an encoder-decoder framework.
- **Generate & Rank** [31]: The pre-trained-model-based MWP solver with equation re-ranking mechanism.
- **HMS** [6]: The MWP solver with a dependency-based module for encoding and an improved GTS decoder.
- **TM-generation** [32]: The MWP solver uses the decoder of Transformer to predict math expression templates, and then fills the missing operators in the predicted templates by the operator identification layer they designed.
- **ELECTRA-GTS** [4]: To compare in the same understanding level of the problem text, we replace the GTS's encoder with the ELECTRA language model.
- **ELECTRA-GRU-xL** [39]: Similarly, we construct an MWP solver combining the ELECTRA language model, GRU decoder, and cross-attention module for comparing. "**xL**" denotes that the decoder has **x** GRU layers.

[1] https://github.com/SCNU203/Math23k

[2] https://github.com/derderking/Ape-clean

[3] https://github.com/sroy9/mawps

- **ELECTRA-TFM-xL** [40]: Similarly, we construct an MWP solver combining the ELECTRA language model and transformer decoder. "**xL**" denotes that the decoder has **x** transformer decoder layers.
- **GSGSF-xL**: The Goal-Driven MWP Solver with Goal Selection and Feedback proposed in this paper. "**xL**" denotes that all MFNs in the decoder are set to **x** layers.

5.2 Implementation details

Our model is implemented on the Ubuntu system using PyTorch and trained on RTX3090. All math expressions of the MWP samples are converted to the corresponding prefix expressions. For the Chinese ELECTRA language model, we use the version pre-trained on an 180G Chinese large corpus [41]. The dimensionalities of all hidden states of the decoder are set to 768. Our model is trained for 80 epochs, and the mini-batch size is set to 32. In terms of the optimizer, we use AdamW [42] with the value of learning rate set to 2×10^{-5} and 1×10^{-3} in the encoder and decoder respectively. The learning rate will be halved when the loss reduction is less than 0.1 times the current loss. For initialization, all the learnable parameters are sampled on the normal distribution $N\left(0, \left(\frac{0.5}{\sqrt{d}}\right)^2\right)$, where $d = 768$ is the dimensionalities of all hidden states. The results are computed through the greedy search.

Finally, we use answer accuracy and equation accuracy as the metrics to evaluate the model. When using answer accuracy, the prediction is considered correct when the calculated value of the predicted expression is equal to the answer. When using equation accuracy, the prediction is considered correct when the predicted expression is the same as the labeled expression. The answer accuracy rate can demonstrate the MWP solving ability of the models. The equation accuracy in the training set can demonstrate the fitting ability of the models.

5.3 Result analysis

Overall result Table 2 reports the answer accuracy of various baseline models and our proposed model on the Math23k, Ape-clean, and MAWPS datasets. As shown in Table 2, firstly, the models containing pre-trained language model (Generate & Rank, TM-generation, ELECTRA-GTS, ELECTRA-GRU, ELECTRA-TFM, and GSFSF) can better solve the MWPs, which confirms that the strong encoding ability of the pre-trained language model brings a big improvement to the model's performance. Next, our model outperforms all baseline models on the Math23k and Ape-clean datasets, which demonstrates that our model is better than other decoders in Chinese MWP solving. In the small English dataset MAWPS, the performance of GSFSF is the same as that of TM-generation and ELECTRA-TFM.

Performance over four decoder In order to verify our point, we train four different decoders with the same encoder (ELECTRA language model). Here is the detailed difference in how these decoders feed the latest result back to the next decoding process:

- **GTS**: As the decoder of ELECTRA-GTS, GTS feeds the hidden vectors (goal vectors) of the parent node and sibling subtree to the next decoding process. It cannot obtain the information of all generated nodes.
- **GRU**: As the decoder of ELECTRA-GRU, GRU adjusts the hidden vector according to the latest result for the next decoding process. It can keep the historical information in the hidden vector, but it is possible to forget the information of the earlier nodes.
- **Transformer**: As the decoder of ELECTRA-TFM, Transformer treats the embedding of the latest result as the new hidden vector and performs self-attention to capture the related history information for decoding the new token. But in the self-attention of each time step, the hidden vector of the same node is unchanged. So it requires multi-layer stacking, that is, utilizing self-attention and feed-forward network (FFN) multiple times to extract information related to the current decoding process.
- **GSF**: As the decoder of GSFSF, GSF has the goal selection operation which is similar to the self-attention for decoding. But the goal selection operation can capture the most related information directly from all generated nodes in each time step. Because all these generated nodes have been always updated by the goal feedback operation according to the latest result in real-time and be provided to the goal selection operation.

First, when GRU is utilized as the decoder, it outperforms GTS on three datasets. This indicates that a linear-structured decoder is not bad for the MWP task. It may be due to the fact that the length of mathematical expressions is not long enough to cause gradient diffusion in RNN. Then, we can find that our model and the transformer decoder perform similarly on the Ape-clean dataset, but there is a gap in the Math23k dataset. This is due to the large number of samples in the Ape-clean dataset, providing more abundant samples for model training. And the difference in performance on the Math23k dataset shows that GSF has better generalization than the Transformer, namely, it requires fewer samples to learn the mathematical relationship in MWP. In the English dataset MAWPS, the Transformer and GSF achieve the same answer accuracy.

Table 2 Answer accuracy of our model and baseline models

Models	Math23k	Ape-clean	MAWPS
MathEN	0.667	0.734	0.692
GroupAtt	0.695	0.753	0.761
AST-Dec	0.690	0.732	0.756
GTS	0.756	0.798	0.778
Graph2Tree	0.774	–	0.837
SAUSolver	0.751	0.799	0.780
Generate & Rank	0.854	–	0.840
HMS	0.761	0.801	0.803
TM-generation	0.853	–	0.852
ELECTRA-GTS	0.835	0.850	0.789
ELECTRA-GRU-2L	0.859	0.861	0.851
ELECTRA-GRU-4L	0.850	0.855	0.843
ELECTRA-GRU-6L	0.844	0.853	0.843
ELECTRA-TFM-2L	0.855	0.863	0.847
ELECTRA-TFM-4L	0.858	0.863	0.848
ELECTRA-TFM-6L	0.863	0.865	0.852
GSGSF-2L	0.874	0.866	0.852
GSGSF-4L	0.869	0.869	0.845
GSGSF-6L	0.871	0.865	0.845

Performance over number of layers The number of layers is a tunable hyperparameter in ELECTRA-GRU, ELECTRA-TFM, and our model. It should be noted that the layer stacking of our model only exists in the MFN module, not in the entire decoder. We vary the number of layers from 2, 4, and 6 for investigating the effect of the number of layers on the model's performance.

The number of layers of the model can reflect the complexity of the model to a certain extent. When the number of layers increases, the fitting ability of the model tends to be stronger. According to the principle of Occam's Razor, the optimal complexity of the neural network model is the minimum complexity that can fit the training set. At this time, the model has the best generalization. When the complexity of the model exceeds the optimal complexity, over-fitting often occurs, resulting in the decline of the models' performance on the test set.

The results of the study are shown in Table 2 and Fig. 6. First, whether on Math23k or Ape-clean, the increase in the number of layers in ELECTRA-GRU gradually degrades the performance of the model. It may be due to the fact that 2-layer is already the optimal complexity of ELECTRA-GRU. Second, the answer accuracy of ELECTRA-TFM rises slowly when the number of layers increases, which indicates that the Transformer decoder will approach its optimal complexity only when its complexity is high enough. However, when the number of layers is 2,4,6, the number of parameters of the Transformer decoder is 19M, 38M, and 57M respectively, and the enhancement of accuracy brought by the increase of a large number of parameters is weak. Finally, the layer stacking of the GSFSF decoder exists only in MFN, and the model achieves the highest accuracy on Math23k and Ape-clean when the number of layers is 2 and 4, respectively. When the number of layers is 2 and 4, the number of parameters of the GSFSF decoder is 27M and 44M respectively. At this time, GSFSF not only has fewer parameters than the 6-layer Transformer decoder but also achieves higher accuracy.

Ablation study The Goal Selection and Feedback and MFN are central to our decoder structure. To investigate their effectiveness, we conduct several ablation experiments on GSFSF-2L. Table 3 shows the results of the experiment, where "w/o Goal Feedback" denotes that the goal feedback operation is removed. "w/o Goal Selection" denotes that the kth solution vector $\mathbf{r}_k$ is treated as the $(k+1)$th goal vector $\mathbf{q}_{k+1}$, and then use it to perform the $(k+1)$th step token prediction. "w/o MFN" denotes that MFN is replaced with the Feed-Forward Network (FFN) from Transformer, which is the 2-layer perceptron containing ReLU activation function and layer normalization layer.

From Table 3 we can see that the lack of goal selection operation implies that our decoder degenerates to linear-structured RNN, but still achieves an answer accuracy of 0.855 and exceeds the ELECTRA-GTS on Math23k. Second, the lack of goal feedback operation implies that

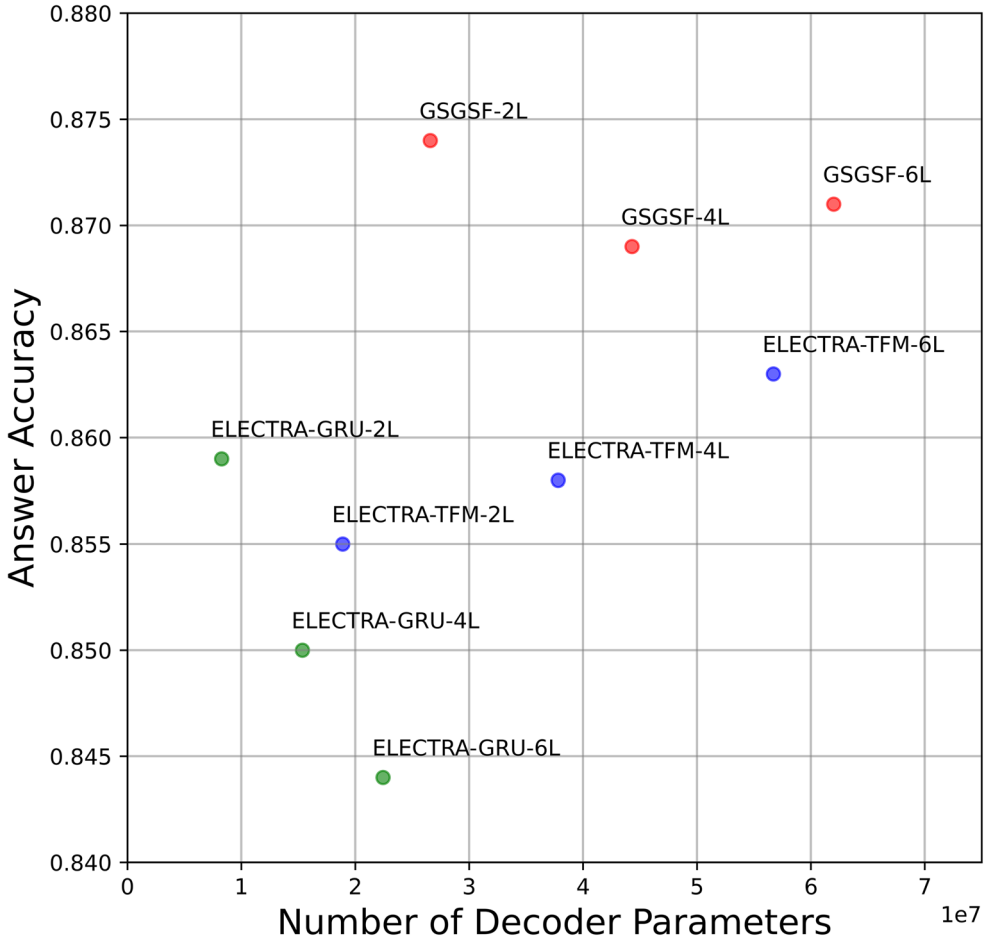


Fig. 6 Performance over number of layers in dataset Math23k

the decoder only uses goal selection operation for decoding, achieving better performance than the linear-structured decoder. Then, the original configuration of the decoder has the highest accuracy, which suggests that the goal feedback operation is helpful and complementary to the goal selection operation. Finally, it can be observed that there is a drop in accuracy when MFN is replaced with the FFN, which shows that MFN does help to generate better hidden state.

Performance on expression length In order to compare the decoding ability of each decoder in more detail, we compute their answer accuracy on each expression length interval (prefix form) separately. The accuracy of models for each math expression interval on the Math23k test set, the Ape-clean test set and the MAWPS dataset are given in Tables 4, 5 and 6 respectively. Five-fold cross-validation is used on the MAWPS dataset, so each sample will act as a test sample once during validation. Therefore, Table 6 shows the answer accuracy on all samples in the MAWPS dataset when they act as the test samples.

In the test set, We can see that the answer accuracy decreases as the length of expression increases. It is in line with the intuition that a longer math expression usually implies a higher complexity of the problem, and the proportion of the training samples in these intervals (over 7) is small. Second, GSF and Transformer outperform the GTS in all situations of different expression length sizes, which indicates that we can better model the mathematical relationship of MWP without a tree-structured neural network. Next, we can see that Transformer performs slightly better than GSF on samples with expression lengths between 3 and 9, and GSF achieves the highest answer accuracy on samples with expression lengths over 9. This indicates that the Transformer is more suitable for

Table 3 Answer accuracy of various decoder configurations

	Math23k	Ape-clean	MAWPS
Configuration	GSFSF-2L	GSFSF-4L	GSFSF-2L
Original	0.874	0.869	0.852
w/o Goal Selction	0.855	0.860	0.847
w/o Goal Feedback	0.863	0.866	0.849
w/o MFN	0.865	0.866	0.845

Table 4 Answer accuracy over expression length on Math23k test set

Expr. len	Ins. num	Prop%	GTS	GRU	Transformer	GSF
0 - 3	174	17.4%	0.925	0.925	0.925	0.925
3 - 5	522	52.2%	0.898	0.914	0.914	0.929
5 - 7	191	19.1%	0.759	0.817	0.832	0.817
7 - 9	66	6.6%	0.545	0.576	0.576	0.621
9 - 11	34	3.4%	0.500	0.529	0.559	0.558
11+	13	1.3%	0.538	0.692	0.692	0.846

Table 5 Answer accuracy over expression length on Ape-clean test set

Expr. len	Ins. num	Prop%	GTS	GRU	Transformer	GSF
0 - 3	564	23.3%	0.874	0.888	0.881	0.883
3 - 5	1073	44.3%	0.911	0.929	0.936	0.928
5 - 7	449	18.5%	0.833	0.833	0.837	0.860
7 - 9	182	7.5%	0.753	0.736	0.764	0.747
9 - 11	98	4.0%	0.571	0.592	0.561	0.622
11+	56	2.3%	0.357	0.393	0.446	0.500

Table 6 Answer accuracy over expression length on MAWPS dataset

Expr. len	Ins. num	Prop%	GTS	GRU	Transformer	GSF
0 - 3	1305	55.5%	0.889	0.924	0.926	0.926
3 - 5	863	36.7%	0.778	0.869	0.866	0.874
5 - 7	107	4.5%	0.140	0.271	0.299	0.234
7 - 9	34	1.4%	0.206	0.324	0.324	0.353
9 - 11	30	1.3%	0.067	0.133	0.200	0.233
11+	14	0.6%	0.071	0.143	0.071	0.071

Table 7 Equation accuracy over expression length on Math23k training set

Expr. len	Ins. num	Prop%	GTS	GRU	Transformer	GSF
0 - 3	4462	20.1%	0.993	0.995	0.995	0.995
3 - 5	11001	49.6%	0.989	0.993	0.994	0.994
5 - 7	4407	19.9%	0.988	0.990	0.992	0.992
7 - 9	1349	6.1%	0.987	0.996	0.996	0.998
9 - 11	574	2.6%	0.974	0.997	1.000	0.998
11+	369	1.7%	0.908	0.927	0.962	0.995

Table 8 Typical cases translated into english. The expressions between brackets are the corresponding midfix expressions of the models' output

Case 1: Mr. Wang bought n_0 shares of a certain stock at n_1 yuan per share and sold them at n_2 yuan per share. He was required to pay a commission of n_3 of the transaction amount (both purchase and sale), how much did Mr. Wang actually earn from buying and selling stocks this time?

ELECTRA-GTS: $- \times \times n_2 n_0 - 1 n_3 \times n_2 n_0$ ✗ $[(n_2 \times n_0) \times (1 - n_3) - n_2 \times n_0]$

GSGSF: $- \times \times n_2 n_0 - 1 n_3 \times \times n_1 n_0 + 1 n_3$ ✓ $[(n_2 \times n_0) \times (1 - n_3) - (n_1 \times n_0) \times (1 + n_3)]$

Case 2: The ticket price of a handball game is n_0 yuan. The ticket price of a swimming game is n_1 yuan less than the n_2 times the ticket price of a handball game. How much more expensive is the ticket price for a swimming game than that of a handball game?

ELECTRA-GTS: $- \times n_0 n_2 n_1$ ✗ $[n_0 \times n_2 - n_1]$

GSGSF: $- - \times n_0 n_2 n_1 n_0$ ✓ $[(n_0 \times n_2 - n_1) - n_0]$

Case 3: In a parking lot, there are n_0 cars and motorcycles. If there is a total of n_1 wheels altogether, where each car has n_2 wheels and each motorcycle has n_3 wheels, what is the number of motorcycles?

ELECTRA-GTS: $\div - \times n_0 n_2 n_1 n_3$ ✗ $[(n_0 \times n_2 - n_1) \div n_3]$

GSGSF: $\div - \times n_0 n_2 n_1 - n_2 n_3$ ✓ $[(n_0 \times n_2 - n_1) \div (n_2 - n_3)]$

samples with medium expression lengths and our decoder has a greater advantage in handling difficult samples. Surprisingly, on the Math23k test set, GSF performs better on samples with expression lengths over 11 than that on samples with expression lengths between 7 and 11. This may be due to the consistency of the MWP types in the training samples and testing samples with expression lengths over 11. In the MAWPS dataset, the number of samples with expression lengths over 5 is only 185, accounting for only 7.2% of the total. In addition, due to the use of five-fold cross validation, the number of training samples must be multiplied by 4/5, resulting in fewer training samples. All of these make each model perform poorly on these samples.

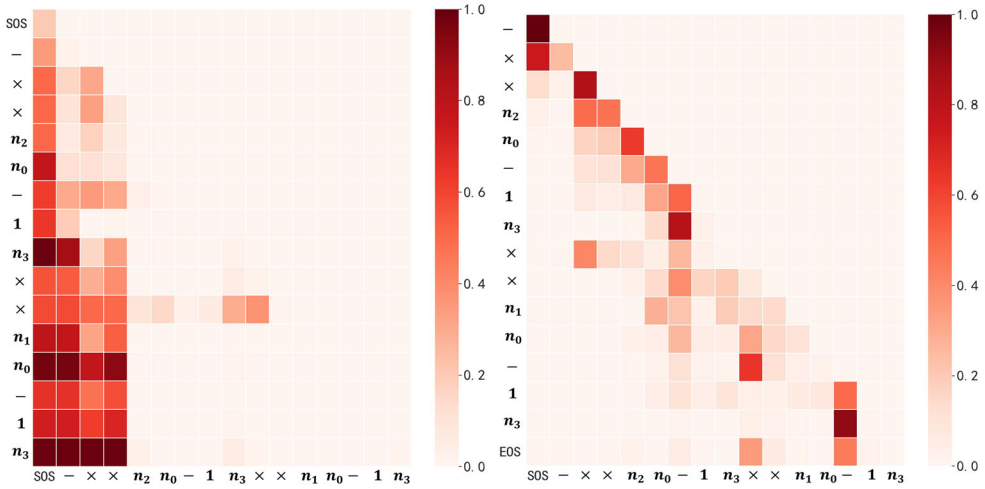
In deep learning, a prerequisite for the model to be able to solve a certain kind of sample is that it can fit such kind of sample. Fitting samples means that the model generalizes the training samples into a variety of templates, and then the model can solve the problems by recalling those templates. Table 7 reports the fit of the model on the Math23k training set, where the equation accuracy in the training set can demonstrate the fitting ability of the models. In our model, the decoder is designed with long-range information acquisition (goal selection operation) and timely information updating (goal feedback operation) mechanisms. When the mechanism of model computation is sufficiently complex, flexible, and not redundant, it is possible to solve more problems by summarizing and memorizing more templates in training. From this point of view, an essential reason why GTS, GRU, and Transformer do not perform as well as GSF on samples with long expressions is that these three models are inferior to GSF in fitting long-expression samples. Namely, GTS, GRU (both with limited historical information acquisition), and Transformer (with only long-range information acquisition mechanism) remembered fewer templates than GSF during training.

5.4 Case study and visualization analysis

Further, we conduct case studies on expressions generated by our model and ELECTRA-GTS and visualization of model decoding in these cases. Three cases are provided in Table 8. Our analyses are summarized as follows:

- From **Case 1**, it can be seen that ELECTRA-GTS generates the correct left subtree "$\times \times n_2 n_0 - 1 n_3$" at the beginning, but generates the wrong right subtree "$\times n_2 n_0$", which demonstrates that the structure of GTS prevents it from getting sufficient information about the previous nodes. Instead, our model is able to obtain sufficient information from previous nodes and finally generates the correct solution for this MWP.
- From **Case 2**, it can be seen that the first goal generated by ELECTRA-GTS is to find the ticket price of a swimming game, while the first goal generated by our model accurately answers the main question posed by the MWP to find the difference of two ticket price, and our model subsequently solves this MWP correctly. The difference between these two first goals indicates that MFN in our model can generate goal vectors with better representations than the two-layer gated-feedforward network of GTS.
- **Case 3** is the sample presented in the section **Introduction** on which GTS made mistake. We can find that our model solves this problem correctly, which shows that the Goal Selection and Feedback does ameliorate the shortage of GTS.

Figures 7, 8 and 9 are the visualization heatmaps of our model's decoding process in **Case 1**, **Case 2** and **Case 3**



(a) Goal Feedback Operation. (b) Goal Selection Operation.

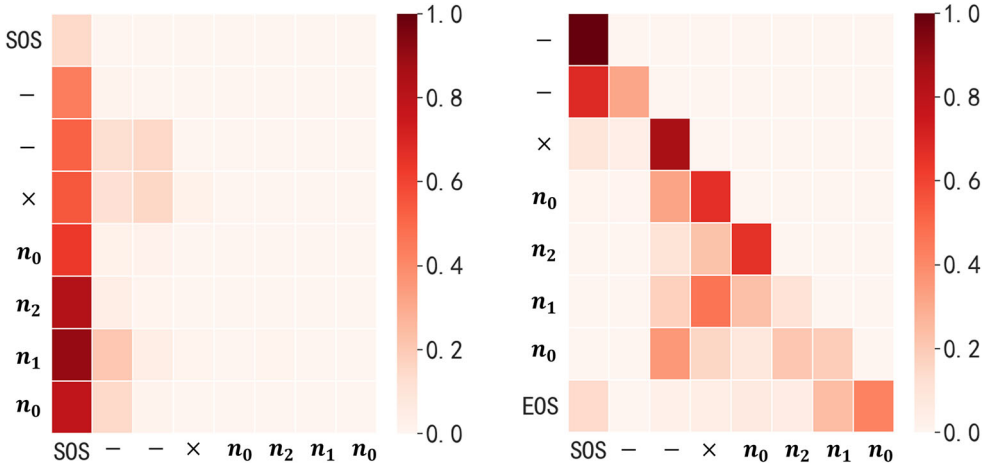
Fig. 7 Visualization of model decoding in case 1

respectively. In the heatmap of Goal Feedback Operation, the shade of the box color indicates the value of the update gate g_i^k, that is, how much information of the last generated node y_k is fed back to the ith goal. In the heatmap of Goal Selection Operation, the shade of the box color indicates the value of the selection weight a_i^k, that is, how much information is selected from the ith goal by the new goal $\mathbf{q}_{k+1}$.

Firstly, it can be seen that the first few goal vectors are updated frequently in Goal Feedback Operation. Because the first few goals are the parent goals of subsequent new sub-goals. When a sub-goal is solved, its parent goal is also solved partially and updated to inform the subsequent decoding process. An interesting point is that after the last math token is generated (the last row in each heatmap of Goal Feedback Operation), all parent goals are completely updated ($g_i^k \approx 1$), and then the ending token "EOS" is generated. Next, in the Goal Selection Operation, the selection range of the new goal often includes its nearby operator, which is its parent node in the binary tree corresponding to the math expression. This shows that the neural model can still learn the tree structure in the math expression without the explicit tree structure.

6 Conclusion

In this paper, we propose a novel decoder that is more suitable for MWP tasks than GTS, especially for long math expressions. Our model uses Goal Selection and



(a) Goal Feedback Operation. (b) Goal Selection Operation.

Fig. 8 Visualization of model decoding in case 2

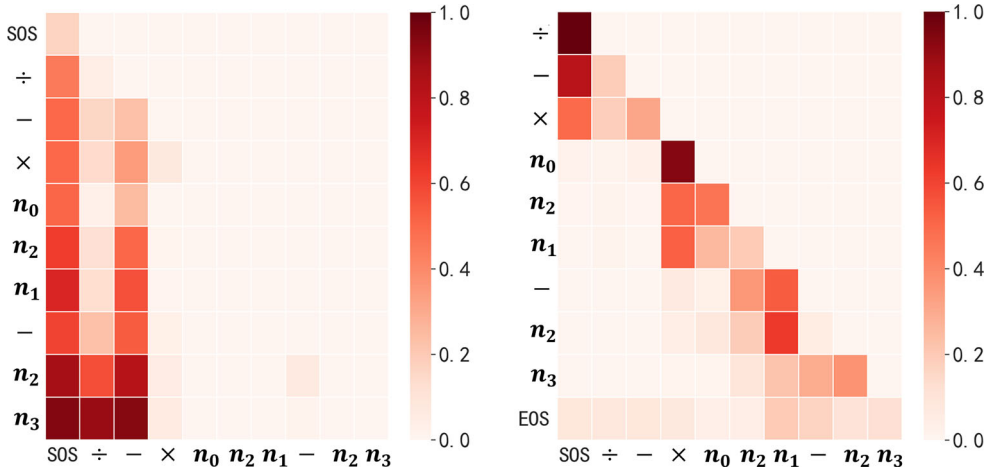


Fig. 9 Visualization of model decoding in case 3

(a) Goal Feedback Operation. (b) Goal Selection Operation.

Feedback and Multilayer Fusion Network for each decoding step, allowing sufficient history information for decoding and better representation for each hidden state. Combining the ELECTRA language model with our decoder, the experimental results demonstrate that our model indeed overcomes the shortcomings of GTS very well and outperforms the previous SOTA systems. For future work, we will focus on improving the generalization of the model to make it perform better on complex samples with small sample sizes.

Acknowledgements This paper is supported by the National Natural Science Foundation of China No. 62177015 and the Natural Science Foundation of Guangdong Province No.2022A1515010148. The source code of this paper is available at: https://github.com/SCNU203/GSFSF

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Wang Y, Liu X, Shi S (2017) Deep neural solver for math word problems. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 845–854
2. Wang L, Wang Y, Cai D, Zhang D, Liu X (2018) Translating a math word problem to a expression tree. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp 1064–1069
3. Wang L, Zhang D, Gao L, Song J, Guo L, Shen HT (2018) Mathdqn: solving arithmetic word problems via deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 32
4. Xie Z, Sun S (2019) A goal-driven tree-structured neural model for math word problems. In: IJCAI, pp 5299–5305
5. Zhang J, Wang L, Lee RK-W, Bin Y, Wang Y, Shao J, Lim E-P (2020) Graph-to-tree learning for solving math word problems. Association for computational linguistics
6. Lin X, Huang Z, Zhao H, Chen E, Liu Q, Wang H, Wang S (2021) Hms: a hierarchical solver with dependency-enhanced understanding for math word problem. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 4232–4240
7. Wu Q, Zhang Q, Fu J, Huang X-J (2020) A knowledge-aware sequence-to-tree network for math word problem solving. In: Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp 7137–7146
8. Clark K, Luong M-T, Le QV, Manning CD (2019) Electra: pre-training text encoders as discriminators rather than generators. In: International conference on learning representations
9. Bobrow DG (1964) Natural language input for a computer problem solving system
10. Slagle JR (1965) Experiments with a deductive question-answering program. Commun ACM 8(12):792–798
11. Fletcher CR (1985) Understanding and solving arithmetic word problems: a computer simulation. Behavior Res Methods Instrum Comput 17(5):565–571
12. Bakman Y (2007) Robust understanding of word problems with extraneous information. Mathematics
13. Yuhui M, Ying Z, Guangzuo C, Yun R, Ronghuai H (2010) Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. In: 2010 Second international workshop on education technology and computer science. IEEE, vol 2, pp 476–479
14. Kushman N, Artzi Y, Zettlemoyer L, Barzilay R (2014) Learning to automatically solve algebra word problems. In: Proceedings of the 52nd annual meeting of the association for computational linguistics (vol 1: long papers), pp 271–281
15. Roy S, Roth D (2017) Unit dependency graph and its application to arithmetic word problem solving. In: Proceedings of the AAAI conference on artificial intelligence, vol 31
16. Shi S, Wang Y, Lin C-Y, Liu X, Rui Y (2015) Automatically solving number word problems by semantic parsing and reasoning. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 1132–1142
17. Huang D, Shi S, Lin C-Y, Yin J (2017) Learning fine-grained expressions to solve math word problems. In: Proceedings of the 2017 conference on empirical methods in natural language processing, pp 805–814
18. Johnson M, Schuster M, Le QV, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado G et al (2017) Google's multilingual neural machine translation system: enabling zero-shot translation. Trans Assoc Computat Linguistics 5:339–351

19. Peris Á, Domingo M, Casacuberta F (2017) Interactive neural machine translation. Comput Speech Language 45:201–220
20. Dabre R, Chu C, Kunchukuttan A (2020) A survey of multilingual neural machine translation. ACM Comput Surveys (CSUR) 53(5):1–38
21. Kong T, Sun F, Liu H, Jiang Y, Li L, Shi J (2020) Foveabox: beyound anchor-based object detection. IEEE Trans Image Process 29:7389–7398
22. Zhao Z-Q, Zheng P, Xu S-T, Wu X (2019) Object detection with deep learning: a review. IEEE Trans Neural Netw Learn Syst 30(11):3212–3232
23. He W, Wu Y, Xiao J, Cao Y (2021) Mgfpn: enhancing multi-scale feature for object detection. J Intell Fuzzy Syst (Preprint):1–11
24. Peng H, Li J, Wang S, Wang L, Gong Q, Yang R, Li B, Philip SY, He L (2019) Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification. IEEE Trans Knowl Data Eng 33(6):2505–2519
25. Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D (2019) Text classification algorithms: a survey. Information 10(4):150
26. Chen Z, Ren J (2021) Multi-label text classification with latent word-wise label information. Appl Intell 51(2):966–979
27. Lin T-E, Xu H (2019) A post-processing method for detecting unknown intent of dialogue system via pre-trained deep neural network classifier. Knowl-Based Syst 186:104979
28. Chen J, Ma J, Wang Y (2019) A survey of human-computer dialogue system based on multiple-round interaction. Nanjing Xinxi Gongcheng Daxue Xuebao 11(3):256–268
29. Zhang X, Zhao X, Tan T (2021) Robust dialog state tracker with contextual-feature augmentation. Appl Intell 51(4):2377–2392
30. Liu Q, Guan W, Li S, Kawahara D (2019) Tree-structured decoding for solving math word problems. 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), pp 2370–2379
31. Shen J, Yin Y, Li L, Shang L, Jiang X, Zhang M, Liu Q (2021) Generate & rank: a multi-task framework for math word problems. In: Findings of the association for computational linguistics: EMNLP 2021, pp 2269–2279
32. Lee D, Ki K, Kim B, Gweon G (2021) Tm-generation model: a template-based method for automatically solving mathematical word problems. J Supercomput. https://doi.org/10.1007/s11227-021-03855-9
33. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization. Stat 1050:21
34. Liang Z, Zhang J, Wang L, Qin W, Lan Y, Shao J, Zhang X (2022) Mwp-bert: Numeracy-augmented pre-training for math word problem solving. In: Findings of the association for computational linguistics: NAACL 2022, pp 997–1009
35. Huang J, Xu M, Zheng H, Shang Q (2021) Chinese math word problems generation network. J Phys Conf Series 2050:012001. IOP Publishing
36. Koncel-Kedziorski R, Roy S, Amini A, Kushman N, Hajishirzi H (2016) Mawps: a math word problem repository. In: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies, pp 1152–1157
37. Li J, Wang L, Zhang J, Wang Y, Dai BT, Zhang D (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, pp 6162–6167
38. Qin J, Lin L, Liang X, Zhang R, Lin L (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), pp 3780–3789
39. Cho K, Van Merrienboer B, Gülçehre Ç, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. In: EMNLP
40. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inform Process Syst, vol 30
41. Cui Y, Che W, Liu T, Qin B, Wang S, Hu G (2020) Revisiting pre-trained models for chinese natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: findings. Association for computational linguistics, pp 657–668. https://www.aclweb.org/anthology/2020.findings-emnlp.58
42. Loshchilov I, Hutter F (2017) Decoupled weight decay regularization

Daijun He received the B.E. degree in computer science and technology from South China Agricultural University, Guangzhou, China, in 2020. Now he is pursuing the M.E. degree with the School of Computer Science, South China Normal University, Guangzhou, China. His research interests include natural language processing and math word problem solving.

Jing Xiao received the B.S. and M.S. degrees in computer science from Wuhan University, Wuhan, China, in 1997 and 2000, respectively, and the Ph.D. degree from the School of Computing, National University of Singapore, Singapore, in 2005. She is currently a Professor with the School of Computer Science, South China Normal University, Guangzhou, China. Her research interests include AI for education, multimedia processing, and data mining.