Don't be Blind to Questions: Question-Oriented Math Word Problem Solving

Zhenwen Liang[1], **Jipeng Zhang**[2], and **Xiangliang Zhang**[✉1]

[1]University of Notre Dame, {zliang6, xzhang33}@nd.edu
[2]Hong Kong University of Science and Technology, jzhanggr@conect.ust.hk

Abstract

Solving math word problems (MWP) is a challenging task for natural language processing systems, as it requires to not only identify and comprehend the problem description within the context, but also to deduce a solution in accordance with the posed question. Previous solvers have been found to prioritize the context over the question, resulting in low performance when solving multiple questions under the same context. In this paper, we present a question-oriented strategy to address this issue and improve the generalizability of MWP solvers. Our approach features an entity-aware encoder that enhances the connection between MWP context and question via entities in established dependency graphs, aiming at obtaining better problem representations. Then, a question-guided decoder is trained using a contrastive learning strategy to enhance the question representations. Empirical evaluations on four benchmarks demonstrate that our method outperforms previous solvers and exhibits a favorable balance between efficacy and efficiency in MWP solving. In addition, our solver is not reliant on any specific pre-trained model and demonstrates seamless compatibility with different pre-trained model backbones.

1 Introduction

Since the 1960s, automated mathematical reasoning has been recognized as a fundamental challenge for computers (Bobrow, 1964), with math word problem (MWP) solving garnering particular interest within the field of natural language processing (NLP). As shown in Table 1, MWPs typically involve mathematical problems elaborated in natural language that are solved through the use of equations, and solving these problems is a demanding task that requires both text comprehension and mathematical reasoning abilities.

Our codes are released at https://github.com/Zhenwen-NLP/QoS_AACL

Context:	Mr. Wang rides a bicycle from home to school at 16 kilometers per hour, and can reach the school in 0.2 hours. He walks 4 kilometers per hour.
Question 1:	How far is Mr. Wang's home from the school?
Solutions:	MWP-BERT: $(16 \times 0.2)/4$ (Wrong) Ground Truth: 16×0.2
Question 2:	How many times faster is Mr. Wang walking than cycling?
Solutions:	MWP-BERT: $16/4$ (Wrong) Ground Truth: $4/16$
Question 3:	How long does it take Mr. Wang to walk to school?
Solutions:	MWP-BERT: $(16 \times 0.2)/4$ (Correct) Ground Truth: $(16 \times 0.2)/4$

Table 1: An example from UnbiasedMWP (Yang et al., 2022) dataset. MWP-BERT (Liang et al., 2022a) failed to solve 2 out of 3 questions with the same context.

At present, the most advanced MWP solvers tend to be either fine-tuned Seq2Tree models (Jie et al., 2022; Liang et al., 2022b; Zhang et al., 2022) or large language models (LLMs) (Wei et al., 2022; Lu et al., 2022; Wang et al., 2022; Chen et al., 2022) prompted with in-context examples. While LLMs may offer higher accuracy in solving these problems, Seq2Tree models have the advantage of requiring fewer parameters, making them a more economical choice. However, current Seq2Tree solvers have been found to prioritize the context over questions. Previous research (Patel et al., 2021; Yang et al., 2022) has demonstrated that these solvers can actually achieve high accuracy on MAWPS (Koncel-Kedziorski et al., 2016) and ASDiv-A (Miao et al., 2020) when the question text has been removed and just using the context, but struggle to achieve similar levels of accuracy on question-diversified datasets such as SVAMP (Patel et al., 2021) and UnbiasedMWP (Yang et al., 2022). For example, as shown in Table 1, it is evident that the state-of-the-art baseline MWP-BERT (Liang et al., 2022a) successfully tackles the most challenging question of the three presented. However, it

Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15–25
November 1–4, 2023.

falls short in addressing the more elementary questions, failing to provide satisfactory answers. The fact that MWP-BERT generates the same solution for both questions 1 and 3 serves as empirical evidence supporting the conclusion that current MWP solvers frequently rely on superficial context-based heuristics rather than devoting sufficient attention to solving the question at hand.

Our approach consists of three components. Firstly, we observe that MWPs often seek to identify an unknown quantity that is related to some known entities. To facilitate the solver's recollection of these entities, we first construct graphs through dependency parsing and then connect common entities that appear in both the context and the question. Although sub-graphs of the dependency tree have been combined with Recurrent-Neural-Network (RNN)-based solvers in Graph2Tree (Zhang et al., 2020) and MultiE/D (Shen and Jin, 2020), it has not been used with pre-trained language model (PLM)-based solvers. Combining dependency graphs and PLM will equip the solver a detailed analysis of the syntactic and semantic dependencies between different elements of the sentence, identifying key nouns and verbs that are essential for understanding the problem. Our PLM-based approach retains the entire dependency tree and also enhances the connection between the context and the question, enabling the solver to effectively parse the MWP, connect relevant information, and avoid potential misunderstandings.

As the second component of our approach, we address the issue of previous Seq2Tree solvers that do not distinguish between the encoded representations of the context and the question. When the integrated representation of context and question is directly fed to the decoder, it tends to prioritize the context and ignore the importance of the question, as the context is usually longer than the question in natural (Patel et al., 2021). To counter this, our approach utilizes only the question to guide the decoding process, resulting in improved question-generalization ability and reduced reliance on the context.

Finally, our approach addresses the difficulties on solving MWPs that may have semantically similar questions but entirely different solutions, which could confuse the solver and lead to incorrect predictions (Liang and Zhang, 2021; Patel et al., 2021). To combat this issue, we introduce a question-contrastive training strategy that helps our solver to focus on the question. While contrastive learning has been widely applied in MWP solver training (Liang and Zhang, 2021; Li et al., 2021; Liang et al., 2022b), most approaches attempt to group analogous MWPs together in the problem representation space. In contrast, our method perturbs the question representation to create negative samples, enabling the model to differentiate genuine questions from these perturbed versions. This helps to mitigate the risk of incorrect predictions due to the confusion caused by semantically similar but ultimately distinct problems.

Overall, we propose a novel Seq2Tree MWP solver named QoS that considers the importance of the question part in MWPs. We conduct extensive experiments on MWP solving over four benchmarks. Our solver not only achieves state-of-the-art accuracy on Math23k (Wang et al., 2017), MAWPS (Koncel-Kedziorski et al., 2016), UnbiasedMWP (Yang et al., 2022), and SVAMP (Patel et al., 2021), to our knowledge, this is also the first fine-tuned approach that beats the chain-of-thought computing approach (Wei et al., 2022) on the MAWPS dataset. Moreover, ablation studies and qualitative analysis are elaborated to prove the effectiveness of our approach. We also implement our solver with different language model backbones, in order to showcase its backbone agnosticism and its potential for further advancement in conjunction with the evolution of pre-trained models.

2 Related Work

2.1 Seq2Seq and Seq2Tree Math Word Problem Solving

Deep learning techniques have garnered widespread popularity as the primary method for solving MWPs due to their efficacy in surpassing traditional statistical rule-based algorithms (Hosseini et al., 2014) and semantic parsing methods (Shi et al., 2015; Huang et al., 2017). Deep neural solver (DNS) (Wang et al., 2017) was the pioneer in solving MWPs via Seq2Seq models with an RNN encoder and RNN decoder. A tree-structured decoder (Liu et al., 2019a; Xie and Sun, 2019) was presented to achieve goal-driven decoding. Li et al. (2019) borrowed the idea from Transformer and applied multi-head attention. Moreover, many pre-processing methods and tricks were also very helpful to Seq2Seq/Seq2Tree solvers, such as number mapping (Wang et al., 2017), equation normalization (Wang et al.,

2018), graph construction (Zhang et al., 2020) and data augmentation (Liu et al., 2020). Some low-resource settings (Alghamdi et al., 2022) such as weak supervision (Hong et al., 2021; Liang et al., 2023b) have also been explored.

In addition to the aforementioned RNN-to-RNN solvers, the advent of transformer-based pre-trained language models (PLMs) has allowed researchers to construct significantly more powerful solvers in these years, such as MWP-BERT (Liang et al., 2022a), REAL (Huang et al., 2021), BERT-CL (Li et al., 2021), Generate&Rank (Shen et al., 2021), Deductive Reasoner (Jie et al., 2022), Analogical solver (Liang et al., 2022b) and so on. PLMs are used in these methods as a problem encoder. A decoder is attached to construct a complete Seq2Tree solver, which can be trained end to end.

Our paper aims to address a common weakness that has plagued previous Seq2Tree solvers, namely their inability to adequately focus on the questions. To remedy this issue, we introduce a question-guided solver that is designed to be more resistant to variations in the formulation of questions. Empirical evaluations demonstrate the efficacy of our approach, with the proposed solver outperforming baseline methods by a substantial margin.

2.2 Large Language Models in Math Word Problem Solving

The ability to reason has long been recognized as a fundamental challenge for large language models (LLMs), with these models' mathematical reasoning capabilities not being fully developed until the emergence of chain-of-thought prompts (Wei et al., 2022). With the help of it, large language models (LLMs) have demonstrated a remarkable level of accuracy in MWP solving, surpassing that of previous fine-tuned models, simply by being provided with a few examples of such problem-solving processes. Interestingly, researchers (Kojima et al., 2022) also find that LLMs are sufficient zero-shot reasoners with the prompt "Let's think step by step." A number of studies have pursued the goal of improving the performance of chain-of-thought, including (Shi et al., 2022; Wang et al., 2022; Chen et al., 2022).

While the aforementioned studies have focused on the improvement of chain-of-thought prompting, our approach takes a different tack by leveraging a fine-tuned Seq2Tree model with significantly fewer parameters (more than $100\times$) than LLMs, yet achieving comparable accuracy. In fact, our solver even outperforms the original chain-of-thought approach on the MAWPS dataset. Additionally, our solver has a more transparent structure and is amenable to further fine-tuning and has the potential to combine with LLMs in a knowledge distillation way such as (Ho et al., 2022; Li et al., 2022; Magister et al., 2022; Shridhar et al., 2022; Liang et al., 2023a). Given these strengths, we believe our work represents a meaningful contribution to the field, providing a powerful Seq2Tree model for MWP solving tasks and advancing the research community's understanding of these problems.

3 Our Approach

3.1 Problem Formulation

The goal of our paper is to develop an MWP solver that can take in a description of a problem, represented as $W = \{w_1, w_2, ..., w_n\}$, and output an equation-shaped solution, denoted as S. In addition, we divide the problem description W into two distinct components: the context represented as $W_c \in \mathbb{R}^a$, and the question represented as $W_q \in \mathbb{R}^b$, where a and b are lengths of the context and the question such that $a+b = n$. In the original dataset, the problem description does not differentiate between these two parts, so we designate the final sentence of the problem as the question and the rest part as the context. There are a few exceptional cases in which there is only a single sentence presented in the problem description, in which case we treat that sentence as a question without any accompanying context. Then, we transform the solution into pre-order traversals of solutions trees to simplify the grammar of the solution (Xie and Sun, 2019). In the solution tree, operator nodes serve as the parents of number nodes, which must always be leaf nodes.

3.2 Model Architecture

Our proposed network is shown in Figure 1, which contains an entity-aware encoder and a question-guided decoder.

Entity-Aware Encoding

The encoder is trained to learn the representation of the problem. Firstly, we input the entire problem W into a pre-trained model to get an initial representation Z. We implement BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and

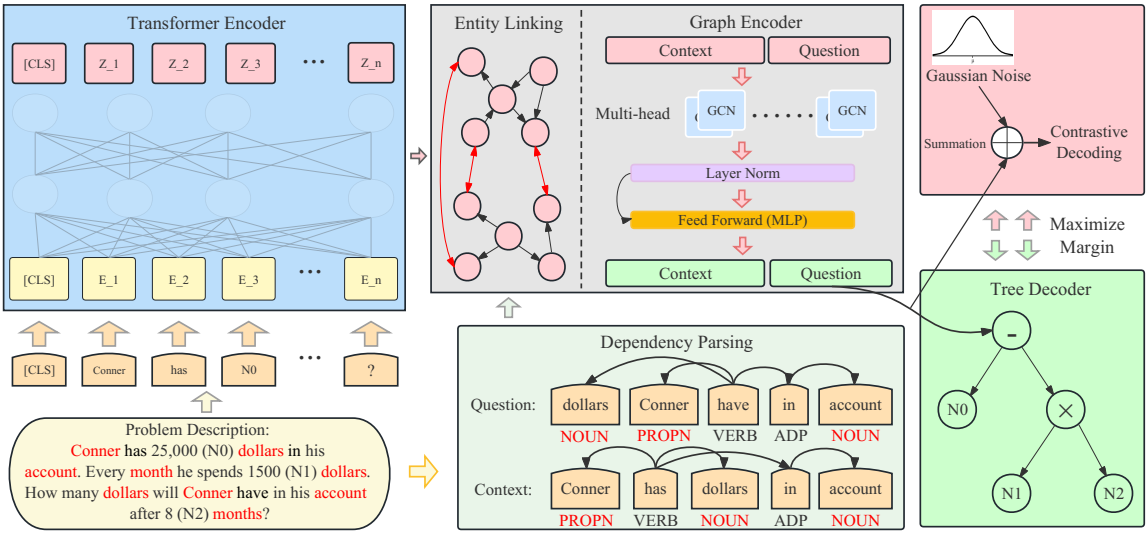

Figure 1: Overview of the proposed QoS model. One MWP is first encoded using a pre-trained language model, followed by the use of dependency graph analysis and entity linking to ensure a high-quality representation of the question. A question-guided decoder is then employed, utilizing contrastive training to maximize the margin between the decoding of the genuine and perturbed question representations.

DeBERTa (He et al., 2020) to show that our solver is generalizable across different PLMs.

Typically, an MWP asks for information about an entity that can be referred to in the context. By identifying the connection between these two references and establishing the understanding that they refer to the same entity, it will construct a more coherent and comprehensive representation of the problem, facilitating the solving process. To this end, we first create a dependency graph of the problem W, which can assist in disambiguating the syntactic roles of words and resolving syntactic ambiguity that may arise in complex word problems. Then we add additional edges related to the entities that appear in both context W_c and question W_q. The construction method of the graph will be explained later in this section. Generally, the workflow of our encoder is:

$$\hat{Z} = enc(W) = Z + GraphNN(Z, \mathcal{A}), \quad (1)$$

where the adjacency matrices of the constructed graphs are represented by $\mathcal{A}$, and our encoder is denoted as enc. Leveraging the context-question association enhanced dependency graph $\mathcal{A}$, we employ a graph neural network, designated as $GraphNN$, to refine the vector $Z \in \mathbb{R}^{h*n}$ where h is the dimension of hidden representations. A residual connection is established between the refined vector and the original vector. The $GraphNN$ is structured in a transformer-like manner, with the Graph Convolutional Network (GCN) block serving as the fundamental building block. Each GCN block can be decomposed into a two-layer graph convolution:

$$GCN(Z, A) = \sigma(\hat{A}(\sigma(\hat{A} \times Z \times X_1))X_2), \quad (2)$$

where $\hat{A}$ is normalized adjacency matrix as mentioned in (Kipf and Welling, 2016), and X_1, X_2 are learnable matrices.

Borrowing from transformers (Vaswani et al., 2017), we also employ a multi-head mechanism. Multiple different GCN blocks are introduced and their output features (equivalent to different heads) are concatenated to get the final representation:

$$MHead(Z, \mathcal{A}) = ||_{h=0}^{H} GCN(Z, \mathcal{A}_{(h \bmod 4)}), \quad (3)$$

where $||$ stands for concatenation operation, and H denotes different heads. $\mathcal{A}$ is a set of adjacency matrices, which represents 4 different methods of graph construction. To achieve this, we first obtain the dependency structure using Stanza (Qi et al., 2020), and then identify all words with POS tags of "NOUN" or "PROPN", as the concepts we are trying to connect are generally nouns. For the first graph, we add edges between w_i and w_j if they refer to the same entity and belong to context W_c and W_q, respectively:

$$\mathcal{A}_0(i,j) = \begin{cases} 1, & \text{if } \ w_i = w_j, w_i \in W_c, w_j \in W_q, \\ 0, & \text{others.} \end{cases} \quad (4)$$

For the second graph, we connect one entity to the neighbors of its another appearance in the dependency graph:

$$\mathcal{A}_1(i,j) = \begin{cases} 1, \text{ if } \mathcal{A}_0(i,k) = 1, j \in NBR(k), \\ 1, \text{ if } \mathcal{A}_0(k,j) = 1, i \in NBR(k), \\ 0, \qquad\qquad \text{others}, \end{cases} \tag{5}$$

where $NBR(k)$ are first-hop neighbors of w_k in the dependency graph. The first graph is utilized to elicit the encoder of the entity's appearance in the context during question encoding, while the second graph is designed to facilitate the bridging of subtle concepts that pertain to the entities. Both $\mathcal{A}_0$ and $\mathcal{A}_1$ are formed by introducing additional edges into the original dependency graph, which can be either directed or undirected. A directed edge only exists from the parent to the child in the dependency tree, while an undirected edge is bidirectional. Therefore, we eventually have four different graphs in total.

Finally, similar to transformers, we add a multi-layer perception (MLP) with Layer Normalization (LN) at the end of our encoder:

$$\begin{aligned} GraphNN(Z,\mathcal{A}) = MHead(Z,\mathcal{A}) + \\ LN(MLP(MHead(Z,\mathcal{A}))). \end{aligned} \tag{6}$$

Dependency parsing can assist in disambiguating the syntactic roles of words, which can be particularly useful in resolving syntactic ambiguity that may arise in complex word problems. Overall, the utilization of these four graphs enhances the capacity of our encoder to comprehend the MWP through dependency parsing, and strengthens the association between the context and the question through additional entity linking.

Question-Guided Decoding

The tree-based decoder (Xie and Sun, 2019) is a popular choice for the design of MWP solvers. It generates the solution in the form of a binary tree, consisting of a predicting module and a decomposing module. Each node in the tree has a hidden state, which the predicting module decodes into a token with an attention score on the problem description. If the prediction is an operator such as $+$ or $-$, the decomposing module splits the node into two children nodes. On the other hand, if the prediction is a quantity, the node becomes a leaf and is not further decomposed. Similar to Recurrent neural networks (RNNs), the hidden state at each time step depends on the previous time step. In this way, we obtain a binary tree whose leaf nodes represent quantities and non-leaf nodes represent operators, which can be easily converted into a mathematical solution. Finally, the decoder outputs the pre-order traversal sequence of the solution tree. However, previously Seq2Tree solvers with tree decoders employed the mean vector of the overall problem representation $\hat{Z}$ denoted in Equation (1), as the hidden state of the root node. However, MWPs often have more context than the question, leading to the mean vector being heavily influenced by the context. This could potentially explain the observation that Seq2Tree solvers tend to converge on simplistic context-based heuristics, as noted in prior research (Patel et al., 2021; Yang et al., 2022). To address the aforementioned issue, we follow the structure of the tree decoder but decompose the problem representation $\hat{Z} \in \mathbb{R}^{h*n}$ into context representation $\hat{Z}_c \in \mathbb{R}^{h*a}$ and question representation $\hat{Z}_q \in \mathbb{R}^{h*b}$ based on the splitting index between W_c and W_q, and use the mean vector of $\hat{Z}_q$ as the initial hidden state of the tree decoder. This seemingly minor modification has been demonstrated to be quite effective in our experiments. This strategy allows the decoder to incorporate a more balanced representation of the problem at the outset, rather than being biased towards the context.

Question-Contrastive Training

Our decoder, as previously mentioned, relies on the learned question representation $\hat{Z}_q$ to guide the solution generation. Given the potential for MWPs to have similar questions but disparate solutions (Liang and Zhang, 2021), it is essential that this question representation be precise. To that end, we introduce question-contrastive learning in our approach, with the aim of augmenting the training process with a degree of variability and bolstering the model's sensitivity to minor data perturbations. This, in turn, provides a more precise encoding for the question and allows for a more accurate decoding towards the solution.

The regular training criteria for the solver is to minimize the negative log probability:

$$\mathcal{L}(W,S) = -\log p(\hat{Z} \mid W) - \log p(S \mid \hat{Z}_c, \hat{Z}_q), \tag{7}$$

where the two terms train the encoder and decoder, respectively. To obtain a question-robust decoder, we apply Gaussian noise to the question representation $\hat{Z}_q$ and get a negative question representation

$\hat{Z}_q^{neg}$:

$$\hat{Z}_q^{neg} = \hat{Z}_q + \lambda \mathcal{N}(0,\, 1), \tag{8}$$

where λ controls the magnitude of the introduced noise. Toward the end of enabling our solver to execute distinct decoding processes for disparate questions, we strive to augment the discrepancy between the decoding based on $\hat{Z}_q$ and $\hat{Z}_q^{neg}$. In other words, we are trying to enlarge the difference $D(\hat{Z}_q, \hat{Z}_q^{neg})$ between those two decoding processes:

$$\begin{aligned} D(\hat{Z}_q, \hat{Z}_q^{neg}) = &- \log p(S \mid \hat{Z}_c, \hat{Z}_q) + \\ &\log p(S \mid \hat{Z}_c, \hat{Z}_q^{neg}). \end{aligned} \tag{9}$$

Then, we adjust our training target to a modified hinge loss with a margin of m:

$$\mathcal{L} = \begin{cases} \mathcal{L}(W, S), & \text{if } m - D(\hat{Z}_q, \hat{Z}_q^{neg}) > 0, \\ -\log p(\hat{Z} \mid W) + m - D(\hat{Z}_q, \hat{Z}_q^{neg}), & \text{otherwise.} \end{cases} \tag{10}$$

When the difference $D(\hat{Z}_q, \hat{Z}_q^{neg})$ exceeds the margin m, which means our model does well on differentiating them, we simply utilize the original training loss as outlined in Equation (7). Conversely, when the discrepancy between the two decoded results falls below the margin m, we alter our decoder training target to augment the difference between them. Eventually, we use loss $\mathcal{L}$ in Equation (10) as our training target, λ and m are hyper-parameters.

4 Experimental Results

4.1 Datasets

Math23k Math23k (Wang et al., 2017) is a commonly utilized benchmark for evaluating the performance of MWP solvers. It is composed of 21,162 Chinese MWPs in the training set and 1,000 in the test set, and is sourced from educational websites.

MAWPS MAWPS (Koncel-Kedziorski et al., 2016) comprises 2,373 English MWPs, which were created by integrating several smaller datasets. In our evaluation, we use a 5-fold cross-validation approach on this dataset due to its small size.

UnbiasedMWP UnbiasedMWP (Yang et al., 2022) consists of 10,264 Chinese MWPs with diverse questions, which are constructed by varying the grounded expressions and annotating corresponding questions by human annotators. An assessment of the solver's performance on this dataset may serve as an indication of its ability to generalize over different questions.

SVAMP The SVAMP (Patel et al., 2021) dataset consists of 1,000 MWPs and is not split into training and testing sets. Previous studies have mainly explored two different settings, i.e., 1). combining MAWPS (Koncel-Kedziorski et al., 2016) and Asdiv-a (Miao et al., 2020) as the training set, with SVAMP as the testing set; 2) splitting SVAMP into 5 folds, and performing a leave-one-out cross-validation, using MAWPS and Asdiv-a as additional training data for each validation. We adopted the latter method, as our proposed approach seeks to optimize the solver towards learning better question generalization capability from the training set to the testing set. In contrast, the first setting does not align with our objective, since the training set has no diversity of questions, while the testing set does. The results in Table 3 demonstrate the ability of our solver, in learning to understand different questions under the same context from the training set and generalizing to the testing set.

4.2 Baselines

GTS (Xie and Sun, 2019) proposes a goal-driven tree-based decoder. **Graph2Tree** (Zhang et al., 2020) utilizes quantity-related graphs to refine the problem representation. **NumS2T** (Wu et al., 2021b) applies explicit numerical encoding. **Multi-E/D** (Shen and Jin, 2020) uses multiple encoders and decoders in MWP solving. **HMS** (Lin et al., 2021) proposes a hierarchical encoder. **EEH-G2T** (Wu et al., 2021a) captures the long-range relationship in the text by graphs. **REAL** (Huang et al., 2021) proposes an augmented learning strategy by retrieving similar MWPs. **BERT-CL** (Li et al., 2021) uses BERT backbone and contrastive learning. **MWP-BERT** (Liang et al., 2022a) presents several pre-training tasks to continually pre-trained a BERT model on MWP corpus. **Deductive Reasoner** (Jie et al., 2022) proposes a deductive decoding method on RoBERTa backbone. **Analogical Solver** (Liang et al., 2022b) is a representation learning method by capturing similar problems during training and uses BERT backbone. **CoT** (Wei et al., 2022) is an in-context learning approach with chain-of-thought prompts on large language models.

4.3 Implementation Details

In this work, we conducted all experiments using an NVIDIA RTX 3090 24G graphics card, implemented in Python using the PyTorch framework. To get the initial representation Z in Equation (1),

we use three different backbones - BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and DeBERTa (He et al., 2020). The training was performed for 100 epochs with a batch size of 16, using the AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2018) optimizer with an initial learning rate of 3e-5 for base models and 6e-6 for large models, which was halved every 30 epochs. The weight decay during training was set to 0.1, and a dropout rate of 0.5 was applied to the decoder to prevent overfitting. During testing, we employed a 5-beam search to improve the quality of the solutions.

4.4 Hyper-parameter Tuning

In order to determine the hyperparameters for our model, we employed a grid search approach with a manually designed search space, using answer accuracy as the evaluation metric. After exploring the search space for the noise magnitude λ, we ultimately selected a value of 1 out of $\{0.01, 0.1, 1, 10\}$. Similarly, for the margin m in Equation 10, we chose a weight of 0.01 from the available options of $\{0.001, 0.005, 0.01, 0.05, 0.1\}$. We set the number of heads H in our graph encoder to 8 out of $\{4, 8, 12, 16\}$. Additionally, we examined the size of the beam search across the range of $\{1, 3, 5, 7\}$ and ultimately selected a search size of 5. Lastly, we selected an embedding size of 512 for the tree-based decoder from the options of $\{128, 256, 512, 1024\}$.

4.5 Backbone Language Models

Our encoder relies on a pre-trained language model to get the representation of MWPs. Specifically, we use three different backbones - BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and DeBERTa (He et al., 2020). For English datasets, three models are cloned from https://github.com/google-research/bert, https://github.com/facebookresearch/fairseq/tree/main/examples/roberta, https://github.com/microsoft/DeBERTa. For Chinese datasets, we use a continually pre-trained model MWP-BERT (Liang et al., 2022a) https://huggingface.co/invokerliang/MWP-BERT-zh as our BERT backbone. Then, we use whole-word-mask (WWM) (Cui et al., 2020) pre-trained Chinese RoBERTa https://github.com/ymcui/Chinese-BERT-wwm and https://github.com/IDEA-CCNL/Fengshenbang-LM as our RoBERTa and DeBERTa backbones, respectively.

	Math23k	MAWPS
GTS	75.6	82.6
Graph2Tree	77.4	83.7
NUMS2T	78.1	-
Multi-E/D	78.4	-
HMS	78.4	80.3
EEH-G2T	78.5	84.8
REAL	82.3	–
BERT-CL	83.2	–
MWP-BERT	84.7	90.1*
Deductive Reasoner	85.1	92.0
Analogical Solver	85.6	91.0*
CoT	–	93.3
Our solver with (base / large) backbone:		
QoS - BERT	**86.0** / 86.7	90.7 / 91.8
QoS - RoBERTa	85.2 / 86.9	**92.1** / **93.4**
QoS - DeBERTa	85.0 / **87.5**	90.6 / 92.3

Table 2: Accuracies on Math23k and MAWPS datasets. * denotes our reproduction. The best base/large models are bolded.

4.6 Accuracy Comparison

We first report the results on two well-acknowledged datasets, Math23k and MAWPS. As shown in Table 2, our solver outperforms baselines on both benchmarks, achieving a state-of-the-art performance. This improvement over Seq2Seq and Seq2Tree baselines on both datasets is statistically significant, as confirmed by a t-test with a p-value < 0.01. While the improvement on MAWPS may appear small, the rest of the problems in this dataset are difficult to solve. It is worth mentioning that our solver with RoBERTa (large) achieves higher accuracy on MAWPS dataset than the CoT approach presented in (Wei et al., 2022), which relies on a 540B-parameter backbone and has a 93.3% accuracy.

To further test the capabilities of our solver, we conduct experiments on two highly challenging benchmarks, UnbiasedMWP and SVAMP. These datasets are created by introducing diverse variations in the MWP questions. Previous Seq2Seq solvers with limited generalization capability have been unable to achieve satisfactory accuracy on these datasets. However, as shown in Table 3, our solver not only becomes the top performer on UnbiasedMWP, but also achieves a superior accuracy on SVAMP. While it is true that LLM-based ap-

	UnbiasedMWP	SVAMP
GTS	63.7	54.6*
Graph2Tree	64.6	58.2*
MWP-BERT	80.1	63.1*
Roberta-Graph2Tree	–	65.0
Deductive Reasoner	82.7*	69.8*
Analogical Solver	82.3*	68.7*
CoT	–	79.0
Our solver with (base / large) backbone:		
QoS - BERT	82.5 / 83.9	50.5 / 69.9
QoS - RoBERTa	**82.8** / 85.1	**60.5** / 72.4
QoS - DeBERTa	82.4 / **86.8**	58.8 / **72.6**

Table 3: Accuracies on UnbiasedMWP and SVAMP datasets. Since the two datasets are recently released and few papers have evaluated their methods on them, only representative baselines (* denotes our reproduction) are selected and compared with our proposed method. The best base/large models are bolded.

Entity-aware encoding	Question-guided decoding	Contrastive Training	Math23k	SVAMP
✗	✗	✗	85.1	65.3
✓	✗	✗	86.0	70.5
✗	✓	✗	85.0	65.2
✗	✗	✓	84.9	65.3
✗	✓	✓	85.5	69.1
✓	✓	✗	86.0	70.1
✓	✗	✓	86.1	70.3
✓	✓	✓	**86.9**	**72.4**

Table 4: Accuracy among different ablated models. All models have RoBERTa-large backbone.

proaches (Wei et al., 2022; Lu et al., 2022; Wang et al., 2022; Chen et al., 2022) have achieved impressive results on the SVAMP dataset, surpassing the accuracy of our solver described here, we posit that our work is fundamentally distinct and not in competition with them. A detailed discussion of this can be found in Section 5.

4.7 Ablation Study

To assess the contribution of each component to the overall performance of the solver, various combinations of these methods are implemented in Table 4. ✗ and ✓ indicate different combinations of the proposed components. To further elaborate, the term "without entity-aware encoding" refers to the condition in which the adjacency matrix of the original dependency graph is utilized solely to form $\mathcal{A}$ in Equation (1). Next, "without question-guided decoding" signifies the scenario in which the entire problem representation is directly input to the decoder. Finally, "without question-contrastive training" refers to the use of Equation (7) as the loss function in place of Equation (10). Generally, the results demonstrate the significance of all three components in achieving optimal performance. To be more specific, the entity-aware encoding can always offer accuracy improvement with different decoding strategy. And the question-guided decoding and question-contrastive loss should be paired to exert their full potential. Because the decoding process is dominated by the context without question decoding, thus question-contrastive training will be less effective. On the other hand, the question decoding without the contrastive loss will be short of obtaining precise representation for similar but different questions (Liang and Zhang, 2021).

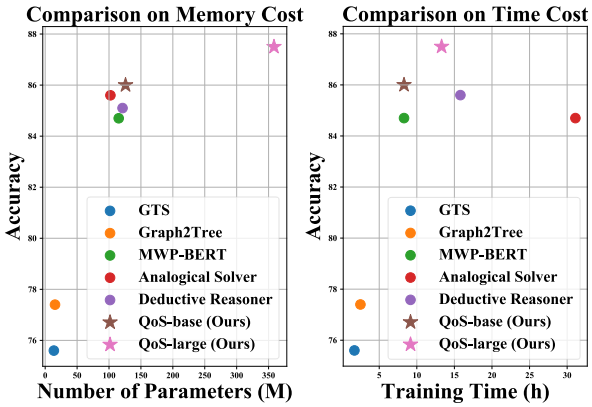

Figure 2: Comparison on memory and time cost between our solver and baselines on Math23k.

4.8 Model Efficiency

As shown in Figure 2, we evaluate the efficiency of our proposed solver by comparing its training time and memory cost with that of representative baselines. The inference time is not considered in our analysis as they all operate at milliseconds scale. We measure the training time cost from the initiation of the training process until the solver reached convergence at its optimal performance. All the results are measured on a single RTX 3090 24G graphics card to ensure fairness and our backbone encoder is RoBERTa. Our results in Figure 2 demonstrate that our model exhibits a satisfying efficiency advantage over existing Seq2Tree solvers while obtaining a superior performance. We can also find that the analogical solver is particularly slow because it needs to seek similar MWPs during training thus requires a much longer time to converge, although it contains a similar number of parameters as ours.

Context:	In a tree planting activity, the fifth-grade students planted 145 trees, 17 fewer than the sixth-grade students. The number of trees planted by the sixth-grade students is 1.5 times that of the fourth-grade.
Question 1:	How many trees did the sixth-grade students plant?
Solutions:	MWP-BERT: $(145 + 17)/1.5$ W/o our encoding: $(145 + 17)/1.5$ W/o our decoding: $145 + 17$ QoS (Ours): $145 + 17$
Question 2:	How many trees did the fourth-grade students plant?
Solutions:	MWP-BERT: $(145 + 17)/1.5$ W/o our encoding: $(145 + 17)/1.5$ W/o our decoding: $(145 + 17)/1.5$ QoS (Ours): $(145 + 17)/1.5$
Question 3:	How many more trees are planted by fifth-grade students than those by fourth-grade students?
Solutions:	MWP-BERT: $145 + 17 - 145$ W/o our encoding: $(145 + 17)/1.5$ W/o our decoding: $(145 + 17)/1.5$ QoS (Ours): $145 - ((145 + 17)/1.5)$
Question 4:	How many times as many trees were planted by fifth-grade students as fourth-grade students?
Solutions:	MWP-BERT: $(145 + 17) * 1.5/145$ W/o our encoding: $145/((145 + 17)/1.5)$ W/o our decoding: $(145 + 17)/145$ QoS (Ours): $145/((145 + 17)/1.5)$

Table 5: Our case study. Correct solutions are colored in green and wrong solutions are in red.

4.9 Case Study

To investigate the effectiveness of our approach, we conducted a case study that compared our solutions to those produced by the baseline MWP-BERT model and two ablated versions of our approach as shown in Table 5. The first ablated model excluded the use of entity linking in the original dependency graph, while the second replaced our decoder with a vanilla tree-decoder (Xie and Sun, 2019). It is worth noting that MWP-BERT misconstrued the first question and seemed to be reasoning based on the context rather than the question itself, whereas our solver accurately solved it. As for the fourth question, our solver utilized entity-aware encoding to precisely comprehend the question's intent and provide a correct solution. This case study demonstrates that our solver is able to concentrate on the questions during the reasoning process and exhibits superior generalizability on questions.

5 Discussion on Large Language Models

In this era of LLMs, it is not surprising that our method cannot surpass the performance of LLMs on MWP benchmarks such as (Wei et al., 2022; Wang et al., 2022; Lu et al., 2022). However, our approach still represents a significant advancement in the field of MWP solving. Its computational efficiency makes it practical for real-world situations where cost and speed are important considerations. For instance, our method has the potential to utilize the capabilities of LLMs through knowledge distillation, serving as a student model to improve performance while maintaining efficiency, like (Ho et al., 2022; Magister et al., 2022; Shridhar et al., 2022; Liang et al., 2023a). Therefore, we believe this work has its value to the research field.

6 Conclusion

In this work, we present a question-oriented strategy to solve math word problems, addressing the limitations of the question-generalizability of previous solvers. Our model contains an entity-aware encoder, a question-guided decoder and question-contrastive loss. Empirical evaluations demonstrate that our approach outperforms fine-tuned Seq2Tree models and even some large language models in terms of effectiveness. Through our case study, we provide further evidence of the effectiveness of our approach. Overall, our solver has a simple structure but exhibits a favorable balance between efficacy and efficiency, and it is also compatible with diverse language model backbones. We believe it has the potential to serve as a valuable reference for future research endeavors in the community.

Limitation

Despite achieving superior question-generalization ability, our solver still encounters challenges in explaining its reasoning steps in a human-like manner. This is a common limitation among Seq2Tree MWP solvers when compared to chain-of-thought approaches. Additionally, while our solver demonstrates state-of-the-art performance among its peers, its accuracy is still lower in comparison to the latest chain-of-thought approaches such as (Wang et al., 2022; Chen et al., 2022).

In our future work, we plan to investigate methods for integrating our approach with large language models (LLMs) in order to harness their exceptional mathematical reasoning capabilities, making our solver both more explainable and more powerful.

References

Reem Alghamdi, Zhenwen Liang, and Xiangliang Zhang. 2022. Armath: a dataset for solving arabic math word problems. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 351–362.

DG Bobrow. 1964. Natural language input for a computer problem solving system. *Ph. D. Thesis, Department of Mathematics, MIT*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. In *EMNLP: Findings*, pages 657–668.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *ICLR*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. In *AAAI*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *EMNLP*, pages 805–814.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. In *Findings of EMNLP*, pages 786–796.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In *ACL*, pages 5944–5955.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *NAACL*, pages 1152–1157.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *ACL*, pages 6162–6167.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2022. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*.

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Zhou, Chao Li, Hongzhi Liu, and Yunbo Cao. 2021. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. *arXiv preprint arXiv:2110.08464*.

Zhenwen Liang, Wenhao Yu, Tanmay Rajpurohit, Peter Clark, Xiangliang Zhang, and Ashwin Kaylan. 2023a. Let gpt be a math tutor: Teaching math word problem solvers with customized exercise generation. *arXiv preprint arXiv:2305.14386*.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022a. Mwp-bert: Numeracy-augmented pre-training for math word problem solving. In *NAACL*, pages 997–1009.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Yan Wang, Jie Shao, and Xiangliang Zhang. 2023b. Generalizing math word problem solvers via solution diversification. In *AAAI*, volume 37, pages 13183–13191.

Zhenwen Liang, Jipeng Zhang, and Xiangliang Zhang. 2022b. Analogical math word problems solving with enhanced problem-solution association. *EMNLP*.

Zhenwen Liang and Xiangliang Zhang. 2021. Solving math word problems with teacher supervision. In *IJCAI*, pages 3522–3528.

Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, and Shijin Wang. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *AAAI*, pages 4232–4240.

Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Reverse operation based data augmentation for solving math word problems. *arXiv preprint arXiv:2010.01556*.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019a. Tree-structured decoding for solving math word problems. In *EMNLP*, pages 2370–2379.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *ICLR*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *ACL*, pages 975–984.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *NAACL*, pages 2080–2094.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *ACL*, pages 101–108.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of EMNLP*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *COLING*, pages 2924–2934.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*, pages 1132–1142.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2022. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS/NeurIPS*, pages 5998–6008.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *EMNLP*, pages 1064–1069.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *EMNLP*, pages 845–854.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021a. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of EMNLP*, pages 1473–1482.

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2021b. Math word problem solving with explicit numerical values. In *ACL*, pages 5859–5869.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305.

Zhicheng Yang, Jinghui Qin, Jiaqi Chen, and Xiaodan Liang. 2022. Unbiased math word problems benchmark for mitigating solving bias. *arXiv preprint arXiv:2205.08108*.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *ACL*, pages 3928–3937.

Wenqi Zhang, Yongliang Shen, Yanna Ma, Xiaoxia Cheng, Zeqi Tan, Qingpeng Nong, and Weiming Lu. 2022. Multi-view reasoning: Consistent contrastive learning for math word problem. *Findings of EMNLP*.