ATHENA: Mathematical Reasoning with Thought Expansion

Anonymous ACL submission

Abstract

Solving math word problems depends on how to articulate the problems, the lens through which models view human linguistic expressions. Real-world settings count on such a method even more due to their lexical sophistication on the same mathematical operations. Earlier works constrain available thinking processes by repeatedly training the patterns or relations between quantities without considering their validity in the context of the problems. We tackle the above challenges and propose **A**ttention-based **TH**ought **E**xpansion **N**etwork **A**rchitecture (ATHENA) to learn mathematics so that it can be practical enough in real-world settings. We introduce thought expansion that maximizes feasible reasoning pathways by mimicking human thinking mechanisms. Thought expansion generates candidate thoughts carrying consistent representation for each mathematical expression and yields reasonable thoughts, filtered by solidly updated reasoning vectors. Our experiments show that ATHENA achieves a new state-of-the-art stage toward the ideal method that is compelling in variant questions even when the informativeness in training examples is restricted. [1]

1 Introduction

Math word problem (MWP) solving is one of the fundamental reasoning tasks of answering a mathematical question by understanding a complex, intricate system of human lexical expressions. Models' ability to solve a problem depends on a method that articulates the problem, the lens through which they view human lexical expressions. Ideal MWP methods produce decent outputs in real-world situations that require more lexically sophisticated on the same mathematical expressions than artificially generated problem sets. For example, "×" can count all elements equally divided in multiple boxes but calculate area from length and width or measure tax fee from the tax rate.

[1]The code will be provided via GitHub once the work is published.

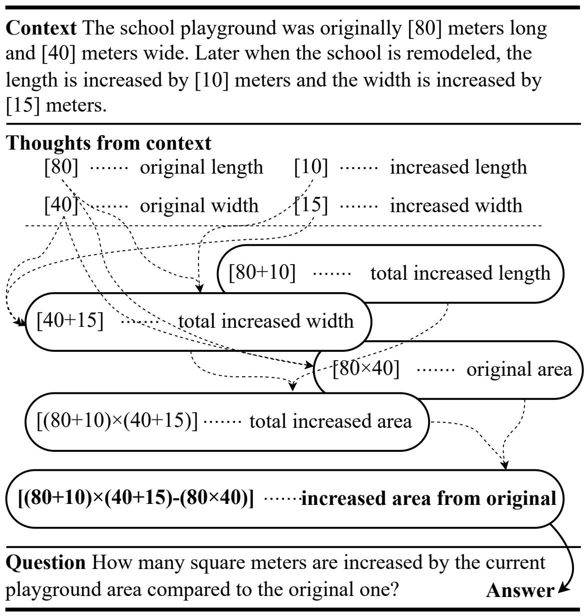

Figure 1: Visualization of thoughts constructed for solving a problem sample from the UnbiasedMWP dataset, one of our benchmarks.

It is significant how we can estimate if the model has learned mathematical reasoning to qualify for the ideal MWP methods. We state that the ideal methods that learn mathematics must be able to solve previously unseen problems if they are applications of mathematical operations that models have already seen or soundly solve problems even when given examples to learn are restricted.

Prior approaches mostly concentrate on enhancing translation by giving problem-level knowledge or quantity relations, not extracting the real-world concepts of individual operations. We observe the following limitations of the approaches. First, they typically stick with one option involving a particular order when generating the structure, leading to constraining other available thinking processes. For example, prior works do not take into account the counter-intuitive approach once they determine to generate a certain mathematical operation as their

Context The school playground was originally [80] meters long and [40] meters wide. Later when the school is remodeled, the length is increased by [10] meters and the width is increased by [15] meters.	
Train on an example of a question-solution pair under the context above.	
Question How many square meters is the original playground area?	**Solution** (80×40)
Test on variant questions that share the context above.	
Q0 How many times the length of the original playground was the width?	
DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (X)	UnbiasedMWP $(80 + 10) \times (40 - 15)$ (X)
UnbiasedMWP (1:N) $80 \div 40$ (O)	UnbiasedMWP (1:N) $80 \div 40$ (O)
Q1 How many square meters is the current playground area?	
DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (X)	UnbiasedMWP $(80 + 10) \times (40 + 15)$ (O)
UnbiasedMWP (1:N) 80×40 (X)	UnbiasedMWP (1:N) $(80 + 10) \times (40 + 15)$ (O)
Q2 How many square meters are increased by the current playground area compared to the original one?	
DeductReasoner	**ATHENA**
UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)	UnbiasedMWP $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)
UnbiasedMWP (1:N) 80×40 (X)	UnbiasedMWP (1:N) $(80 + 10) \times (40 + 15) - (80 \times 40)$ (O)
An example with a lexically similar context to that of above from the UnbiasedMWP	
Context The school basketball court was [20] meters long and [12] meters wide. After the renovation, the length is increased by [8] meters, and the width increases by [3] meters.	
Question How many square meters are increased?	**Solution** $(20 + 8) \times (12 + 3) - (20 \times 12)$

Table 1: Predictions of DeductReasoner (Jie et al., 2022) and ATHENA on a sample that has variant questions while sharing the common context for the problems. The observation above is when models use RoBERTa-large on UnbiasedMWPs.

first step, which limits the reasoning capability in the end. Second, they repeatedly train the patterns or relations among the quantities that appear in the given problem text without confirming their validity with the context of the problem. As a result, although they achieve high performance on some benchmarks, their performances show poor performance on simple elementary-level problems that do not share lexical patterns (Patel et al., 2021).

We tackle the above challenges and propose ATHENA to learn mathematics so that it can be practical enough in real-world settings. Inspired by Johnson-Laird (2008), we focus on thoughts before considering goals to mimic human thinking processes of reasoning. We develop ATHENA using the thoughts within a thought expansion mechanism that is maximizing the feasible reasoning paths. From the novel approach, we generate candidate thoughts that are consistent for each quantity within the thought expansion, and reasonable thoughts that are filtered by solidly updated reasoning vectors and robust regardless of training bias to reach our goal.

Our experiments show that our approach is strong at predicting mathematical expressions requiring their complicated combinations as shown in Table 1. We observe that ATHENA produces a solid performance when the model needs to deal with previously unseen questions. ATHENA is also very compelling to solve variant questions once it has learned one question established from the shared context. From the experimental results, we conclude that ATHENA reaches another new state-of-the-art stage toward the ideal MWP method that we define as the one that can learn mathematical reasoning.

2 Math Word Problem

Math word problem (MWP) solving is a task of answering a mathematical question by understanding natural language descriptions.

2.1 Problem Formulation

Our task of solving math word problems is defined as follows. Each example in the MWP dataset D has a problem sequence S in natural language as input and an equation $\mathcal{E}$ as expected output. D consists of K (problem, question, equation) triples where K is the number of examples:

$$D = \{(S_{(i)}, \mathcal{E}_{(i)})\}_{i=1,\dots,K}.$$

We use a pre-trained language model (PLM) to embed S. Let $P = (t_1, t_2, \dots, t_n)$ denote a tokenized sequence of S where t_i represents each subword token. PLM output of sequence P is denoted as $X = (x_1, x_2, \dots, x_n)$, where x_i represents an embedding vector of each token t_i.

2.2 Related Work

MWP problems have begun with feature engineering via hand-crafted rules or statistical concepts (Bakman, 2007; Hosseini et al., 2014; Mitra and Baral, 2016). Early works have adopted neural network approaches through end-to-end learning strategies such as sequence-to-sequence or sequence-to-tree. They use sequence networks or manipulate the representation with tree or graph templates to generate mathematical equations in a structurally sophisticated manner (Wang et al., 2017; Zhang et al., 2020). Having developed and become accessible to pre-training and transfer learning, many works have promoted their performance with pre-trained language models (Liang et al., 2022; Shen et al., 2021; Yu et al., 2021; Huang et al., 2021). Most of them aim to enhance the encoder with pre-trained embeddings. Recent work has made a key contribution by utilizing additional knowledge such as semantic meaning. Some take advantage of structural information such as hierarchical dependency, formula structure, graph-edge connection information, order relationships among quantities, and more (Lin et al., 2021; Wu et al., 2021; Huang et al., 2020; Zhang et al., 2020). The reasoning extraction method has recently reached decent performance by considering the operation orders and omitting unnecessary steps of creating already generated mathematical sub-expressions (Jie et al., 2022).

3 ATHENA

Attention-based **TH**ought **E**xpansion **N**etwork **A**rchitecture (ATHENA) is an architecture that expands its thoughts to solve the math word problem. Figure 2 illustrates the overall process of ATHENA. ATHENA extracts initial thoughts from PLM and expands them with reasoning to reach the final thought. We first clarify what is a *thought* as a foundational ingredient of our model, and explain the reasoning and goal vectors that measure the thoughts.

Thought. A thought is an embedding of a possible math expression derived from quantities in a problem representing the contextual meaning of the expression. Let θ denote a thought with hidden size H corresponding to an expression $\mathcal{E}(\theta)$. A goal of the model is to find a thought θ^* that satisfies the ground-truth expression $\mathcal{E}^*$:

$$\mathcal{E}(\theta^*) \equiv \mathcal{E}^*.$$

Reasoning Vector. A reasoning vector represents premises to evaluate and filter candidate thoughts in each depth. Let R_d denote a reasoning vector for depth d. We set an initial reasoning vector R_0 with the [CLS] token from the problem descriptions.

Goal Vector. A goal vector plays a role as ground-truth measurement to evaluate if a thought is an appropriate answer to the question. We set a goal vector G with a tokenized embedding of the punctuation mark in the question description.

3.1 Initial Thought

An initial thought is an embedding that carries each quantity representation illustrated in a context or question description. We mask quantities with [MASK] token and obtain the embeddings that capture contextual information from the perspective of corresponding quantities. We denote a set of thoughts in the initial depth by Θ_0:

$$\Theta_0 = \{x_i \mid x_i \in X, t_i \in P, t_i = [\text{MASK}]\}.$$

Certain quantity representations such as π are necessary for generating mathematical expressions despite not being presented in the contexts or questions. We collect them from a training set and randomly initialize their embeddings. We also put their embeddings to initial thoughts Θ_0.

3.2 Thought Expansion

In each depth, thought expansion constructs candidate thoughts Θ_d and filters them to obtain the *reasonable* thoughts Θ_d^*. Reasonable thoughts are the waypoint thoughts to reach the final thought.

The two stages in a thought expansion are: (1) our model generates candidate thoughts Θ_d from previous thoughts Θ_{d-1}^* through the operations and (2) it reasons about the candidates if they are worth to be reasonable thoughts Θ_d^*. Expansion keeps going until finding one of the reasonable thoughts qualified to be a final thought θ^*.

3.2.1 Candidate Thought

Our model generates a set of possible new thoughts Θ_d from the previous thoughts Θ_{d-1}^* as the candidates. A new thought θ' is a thought of a math expression that two previous thoughts $\theta_i, \theta_j \in \Theta_{d-1}^*$ combine with an arithmetic operation:

$$\mathcal{E}(\theta') = \mathcal{E}(\theta_i) \circ \mathcal{E}(\theta_j) \text{ where } \circ \in \{+, -, \times, \div\}.$$

To make a new thought, we introduce two operation layers whose combination can represent the

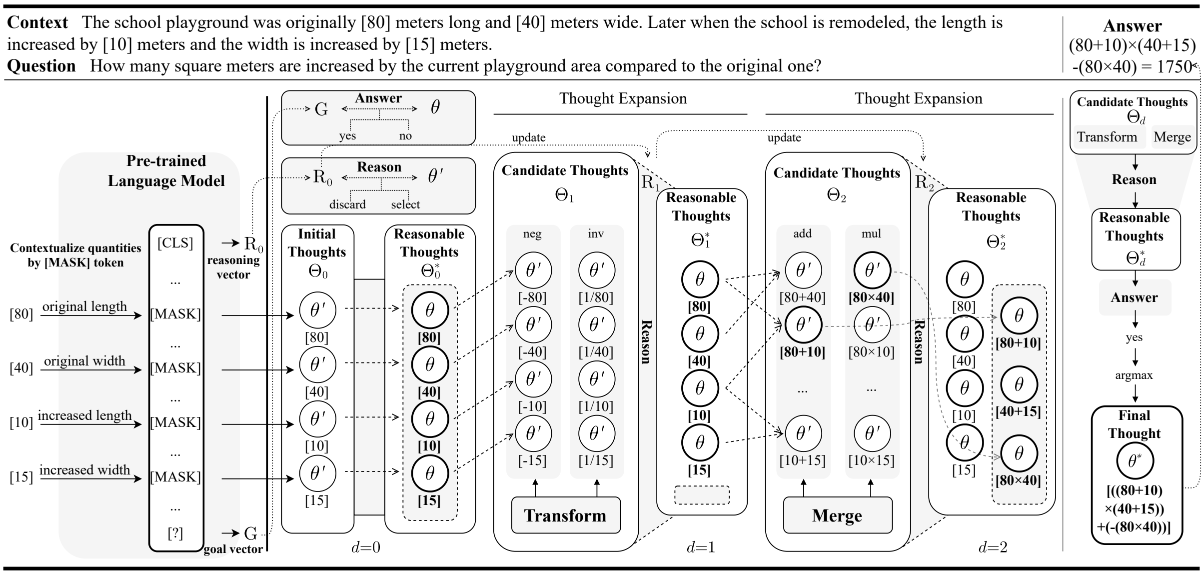

Figure 2: Overall process of ATHENA. First, extract initial thoughts, an inital reasoning vector, and a goal vector from PLM. Second, expand thoughts by transform ($d = 1, 3, 5, \dots$) or merge ($d = 2, 4, 6, \dots$) and generate candidate thoughts. Third, reason on candidate thoughts and obtain new reasonable thoughts. Last, give the reasonable thoughts to the next expansion. Repeat until meeting a thought that answers the goal vector.

arithmetic operations: merge M and transform T. These layers aim to maximize the reflection of the characteristics of arithmetic operations rather than the separate layers of individual arithmetic operations. The definitions of merge and transform are shown below.

Merge. Merge layer M merges a pair of thoughts (θ_i, θ_j) into a new thought θ' such that $\mathcal{E}(\theta')$ applies addition and multiplication to $\mathcal{E}(\theta_i)$ and $\mathcal{E}(\theta_j)$:

$$\overset{\mathrm{op}}{\mathrm{M}} : \theta_i, \theta_j \mapsto \theta'$$
$$\text{s.t. } \mathcal{E}(\theta') = \mathrm{op}(\mathcal{E}(\theta_i), \mathcal{E}(\theta_j)) \text{ where } \mathrm{op} \in \{+, \times\}.$$

Transform. Transform layer T transforms a thought θ into a new thought θ' such that $\mathcal{E}(\theta')$ applies inverse operations of addition and multiplication to $\mathcal{E}(\theta)$:

$$\overset{\mathrm{op}}{\mathrm{T}} : \theta \mapsto \theta'$$
$$\text{s.t. } \mathcal{E}(\theta') = \mathrm{op}(\mathcal{E}(\theta)) \text{ where } \mathrm{op} \in \{-\cdot, \cdot^{-1}\}.$$

We use FeedForward Network (FFN) and multi-head attention inspired by Vaswani et al. (2017) for the implementation of the operation layers. We use FFN referred to as FF for transform layer T. Using multi-head self-attention A_{self} and layer normalization ℓ, we implement merge layer $\mathrm{M}(\theta_i, \theta_j)$ followed by:

$$\mathrm{M}(\theta_i, \theta_j) = \mathrm{FF}(\theta_i + \theta_j + \ell(\mathbf{1}_2^\top \underset{\text{self}}{\mathrm{A}}([\theta_i; \theta_j]))W + b)$$
$$\text{where } W \in \mathbb{R}^{H \times H}, b \in \mathbb{R}^H.$$

This implementation satisfies M^{op} to be commutative for $\mathrm{op} \in \{+, \times\}$:

$$\overset{\mathrm{op}}{\mathrm{M}}(\theta_i, \theta_j) = \overset{\mathrm{op}}{\mathrm{M}}(\theta_j, \theta_i) \text{ and}$$
$$\mathcal{E}(\overset{\mathrm{op}}{\mathrm{M}}(\theta_i, \theta_j)) = \mathcal{E}(\overset{\mathrm{op}}{\mathrm{M}}(\theta_j, \theta_i)).$$

We apply transform layer T for depth $d = 2n - 1$ and merge layer M for depth $d = 2n$ to generate the candidates. In the case of the beginning depth $d = 0$, we use the initial thoughts Θ_0 as the candidates.

3.2.2 Reasonable Thought

Our model iteratively yields reasonable thoughts Θ_d^* that constitute the final thought θ^*. It reasons to select reasonable thoughts from the candidates Θ_d.

Reason. Our model reasons on the candidate thoughts Θ_d to evaluate if they are reasonable by calculating the correlation between the reasoning vector R_d and each thought $\theta \in \Theta_d$ using multi-head attention $\mathrm{A}(Q, K = V)$ and feed-forward network FF with sigmoid σ:

$$\mathrm{reason}(\mathrm{R}_d, \theta) = \sigma(\mathrm{A}(\mathrm{FF}(\theta), \mathrm{R}_d)W_r + b_r)$$
$$\text{where } W_r \in \mathbb{R}^{H \times 1}, b_r \in \mathbb{R}.$$

A thought θ is reasonable if $\mathrm{reason}(\mathrm{R}_d, \theta)$ exceeds a fixed threshold t_r. The reasonable thoughts Θ_d^* become the input of the next iteration and keep proceeding with the thought expansion.

Algorithm 1 Thought Expansion Process of ATHENA

Input: $\Theta_0, \mathrm{R}_0, \mathrm{G}$
Output: $\mathcal{E}^*$
  $d \leftarrow 0$
  $\Theta_0^* \leftarrow \{\theta \mid \theta \in \Theta_0, \mathrm{reason}(\mathrm{R}_0, \theta) \geq t_r\}$
  while $d \leq D$ **or** $\exists \theta \in \Theta_d^*(\mathrm{answer}(G, \theta) > t_f)$ **do**
    $\mathrm{R}_{d+1} \leftarrow \mathrm{R}_d \parallel \mathrm{A}(\mathrm{FF}([\Theta_d^*]), \mathrm{R}_d)$
    $d \leftarrow d + 1$
    if $d = 1, 3, 5 \dots$ **then**
      $\Theta_d \leftarrow \bigcup_{\mathrm{op} \in \{-_{\cdot}, \cdot^{-1}\}} \{\mathrm{T}^{\mathrm{op}}(\theta) \mid \theta \in \Theta_{d-1}^*\}$
    else if $d = 2, 4, 6 \dots$ **then**
      $\Theta_d \leftarrow \bigcup_{\mathrm{op} \in \{+, \times\}} \{\mathrm{M}^{\mathrm{op}}(\theta_i, \theta_j) \mid \theta_i, \theta_j \in \Theta_{d-1}^*\}$
    end if
    $\Theta_d^* \leftarrow \Theta_{d-1}^* \cup \{\theta \mid \theta \in \Theta_d, \mathrm{reason}(\mathrm{R}_d, \theta) \geq t_r\}$
  end while
  $\theta^* \leftarrow \arg\max_{\theta \in \Theta_d^*} \mathrm{answer}(G, \theta)$
  return $\mathcal{E}(\theta^*)$

Update. Our model updates the reasoning vector that was initially created or was given by the previous iteration, with the reasonable thoughts Θ_d^* obtained in the current depth d. We gain the updated reasoning vector for the next depth R_{d+1}, by concatenating all reasonable thoughts Θ_d^* after multi-head attention layer A applied in reason:

$$\mathrm{R}_{d+1} = \mathrm{R}_d \parallel \mathrm{A}(\mathrm{FF}([\Theta_d^*]), \mathrm{R}_d).$$

3.3 Final Thought

A final thought θ^* is the answer to the question. When the thought expansion process finishes, our model decides the final thought by selecting a thought with the maximum score. We have two criteria to terminate the iteration; (1) when the depth reaches the maximum expansion depth D; (2) if there is a thought with the score that exceeds a confidence threshold t_f on iteration. We calculate the score of each reasonable thought $\theta \in \Theta_d^*$ using the multi-head attention A and feed-forward network FF with the goal vector G, activated by sigmoid σ:

$$\mathrm{answer}(\mathrm{G}, \theta) = \sigma(\mathrm{A}(\mathrm{FF}(\theta), \mathrm{G})W_a + b_a)$$
$$\text{where } W_a \in \mathbb{R}^{H \times 1}, b_a \in \mathbb{R}.$$

A thought with the maximum score in the reasonable thoughts becomes a final thought θ^*:

$$\theta^* = \operatorname*{arg\,max}_{\theta \in \Theta_d^*} \left(\mathrm{answer}(G, \theta)\right).$$

Our model bestows the final thought the fidelity to shape the answer to the target question. In summary, we show the overall process to reach the final thought θ^* in Algorithm 1.

4 Experiments

We conduct experiments across a comprehensive range of math word problem (MWP) solving tasks to show that ATHENA outperforms strong baselines in both full datasets and variant versions of the original datasets while being more interpretable in terms of intermediate steps toward the answers.

4.1 Experimental Setups

Baselines. We select four representative approaches as the baselines to compare with ATHENA: Transformer (Vaswani et al., 2017)[2], a goal-driven tree-structured model (GTS) (Xie and Sun, 2019), Graph-to-Tree (Zhang et al., 2020)[3] and DeductReasoner (Jie et al., 2022).[4] Transformer is a sequence-to-sequence approach that uses multi-head attention mechanism while GTS is a strong baseline of sequence-to-tree model. Graph-to-Tree is another approach that adds a graph encoder on top of GTS. We adopt DeductReasoner as an additional baseline that introduces a complex relation extraction method for deductive steps and hence achieves the state-of-the-art performance.

Implementation Details. We use RoBERTa-base and RoBERTa-large as our base pre-trained embeddings (Liu et al., 2019) and Chinese-RoBERTa (Cui et al., 2019) for Chinese benchmarks to compare our baselines. We use pre-layer normalization (Xiong et al., 2020) for our multi-head attention method to fully leverage a dynamic range of embeddings. We set $t_r = 0.5, t_f = 0.95$ and train our model by giving ideal accepted prior thoughts Θ_{d-1}^* and labels of reason and answer in each depth to calculate the loss with binary cross entropy over all labels.[5] Our experiments are performed with Nvidia RTX 3090 GPU.

Dataset. We test the benefits of ATHENA on standard MWP benchmarks that are known as classic and relatively new benchmarks that contain various linguistic expressions in contexts or

[2]We follow hyperparameters by Lan et al. (2022) for both vanilla transformer and RoBERTa-based transformer.

[3]We follow the best hyperparameter settings in Patel et al. (2021) for both vanilla models and RoBERTa-based models.

[4]We use their hyperparameter setups. We use the MAWPS setup for testing ASDiv-A, and use the Math23k setup for UnbiasedMWP. Since the authors do not provide setups for RoBERTa-large, we optimize the model and report the best score with half batch size and half learning rate from those used in the RoBERTa-base setup.

[5]We explain detailed training settings and hyperparameters in Appendix A

	MAWPS	ASDiv-A	Math23k	SVAMP	UnbiasedMWP	SVAMP (1:N)	UnbiasedMWP (1:N)
Language	English	English	Chinese	English	Chinese	English	Chinese
Random embedding							
Transformer	85.6	59.3	61.5	20.7	$20.5_{\pm 0.73}$	$9.7_{\pm 0.19}$ (14.9)	$16.9_{\pm 0.31}$ (51.5)
GTS	82.6	71.4	75.6	30.8	$26.2_{\pm 0.20}$	$12.2_{\pm 0.37}$ (43.8)	$22.8_{\pm 0.22}$ (65.0)
Graph-to-Tree	83.7	77.4	77.4	36.5	$27.2_{\pm 0.37}$	$25.3_{\pm 0.12}$ (52.5)	$24.3_{\pm 0.25}$ (66.4)
RoBERTa-base							
R-Transformer	88.4	72.1	76.9	30.3	$18.3_{\pm 0.15}$	$13.5_{\pm 0.33}$ (33.4)	$14.9_{\pm 0.20}$ (53.1)
R-GTS	88.5	81.2	-	41.0	-	$40.9_{\pm 0.50}$ (64.4)	-
R-Graph-to-Tree	88.7	82.2	-	43.8	-	$31.8_{\pm 0.36}$ (66.7)	-
DeductReasoner	$92.0_{\pm 0.20}$	$83.1_{\pm 0.24}$	**85.1**$_{\pm 0.24}$	$45.0_{\pm 0.10}$	$31.6_{\pm 0.51}$	$42.5_{\pm 0.41}$ (69.1)	$26.5_{\pm 0.55}$ (79.5)
ATHENA(Ours)	**92.2**$_{\pm 0.10}$	**86.4**$_{\pm 0.11}$	$84.4_{\pm 0.24}$	**45.6**$_{\pm 0.50}$	**36.2**$_{\pm 0.67}$	**52.5**$_{\pm 0.50}$ (70.1)	**35.4**$_{\pm 0.45}$ (80.5)
RoBERTa-large							
DeductReasoner	$92.6_{\pm 0.16}$	$89.1_{\pm 0.46}$	$85.8_{\pm 0.42}$	$50.3_{\pm 0.30}$	$34.9_{\pm 0.11}$	$51.6_{\pm 0.38}$ (75.4)	$33.7_{\pm 0.60}$ (83.2)
ATHENA(Ours)	**93.0**$_{\pm 0.20}$	**91.0**$_{\pm 0.13}$	**86.5**$_{\pm 0.25}$	**54.8**$_{\pm 0.63}$	**42.0**$_{\pm 0.57}$	**67.8**$_{\pm 0.58}$ (79.8)	**48.4**$_{\pm 0.38}$ (84.8)

Table 2: Comparison of MWP methods. We use MAWPS, ASDiv-A, and Math23k for standard evaluation, SVAMP and UnbiasedMWP to evaluate the ability to solve entirely unseen, various expressions, and SVAMP and UnbiasedMWP with the one-to-many test to estimate the adaptability of confusing linguistic subtlety.

questions for mathematical reasoning. The standard benchmarks are **MAWPS** (Koncel-Kedziorski et al., 2016), **ASDiv-A** (Miao et al., 2020), and **Math23k** (Wang et al., 2017). MAWPS is an English corpus collected from the online math word problem repository, and Math23k is a Chinese corpus crawled from online posts. ASDiv-A is an acronym of An arithmetic subset of Academia Sinica Diverse dataset (ASDiv-A), consisting of diverse English lexical patterns.

The relatively new benchmarks either alter the standard benchmarks or vary the grounded expressions from the collected data to evaluate the model performance without bias from learned data. **SVAMP** (Patel et al., 2021) varies in the components of one of the standard benchmarks, ASDiv-A to evaluate various contextual expressions on elementary-level arithmetic problems.**UnbiasedMWP** (Yang et al., 2022) is an online-crawled Chinese corpus that augments the questions from the same context to evaluate models if they are able to generate adequate corresponding mathematical expressions. We split MAWPS, ASDiv-A, Math23k, SVAMP, following Jie et al. (2022) and Patel et al. (2021), respectively.

One-to-Many Test. We conduct one-to-many variants tests to measure model generalization to many variant questions from one example within the common context. We select two datasets SVAMP and UnbiasedMWP to apply for this test. Each example in the dataset has a problem sequence that is composed of context and question descriptions. Within the groups by context, we split the examples one-to-many. One example per group goes to a training set while the multiple examples move to a test set. We use examples that do not have other variants within the context group as a validation set. We name the resorted SVAMP and UnbiasedMWP using the one-to-many setup as SVAMP(N:1) and UnbiasedMWP(N:1).

4.2 Results

We repeat our experiments 5 times with different random seeds and report the average answer accuracy with the standard error. We report results on multiple benchmarks, variants splitting tests, the impact of pre-trained language models depending on their size, and ablation tests.

Overall Performance. Table 2 shows the performance of different methods on 7 benchmarks. As seen from the table, ATHENA establishes new state-of-the-art results for overall benchmarks. ATHENA outperforms prior MWP methods on all occasions with one exception of its performance on Math23k when trained on the RoBERTa-base model. When compared to the most competitive work DeductReasoner, ATHENA obtains a relative improvement of 3.84%p on total benchmarks.

Performance on One-to-Many Test. We note that ATHENA achieves huge performance gains compared to the second-best method, from 42.5% to 52.4% and from 26.5% to 35.0% on SVAMP (1:N) and UnbiasedMWP (1:N), respectively. As illustrated in section 4.1, we evaluate our model on SVAMP (1:N) by training with one example per problem set to test how well ATHENA reasons on the questions that use the same textual descriptions

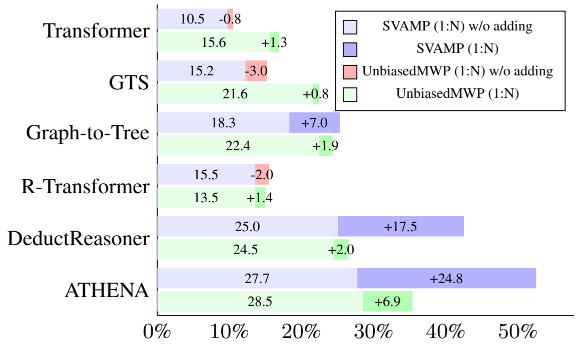

Figure 3: Accuracy changes when adding one example per context into the training set by applying the one-to-many test.

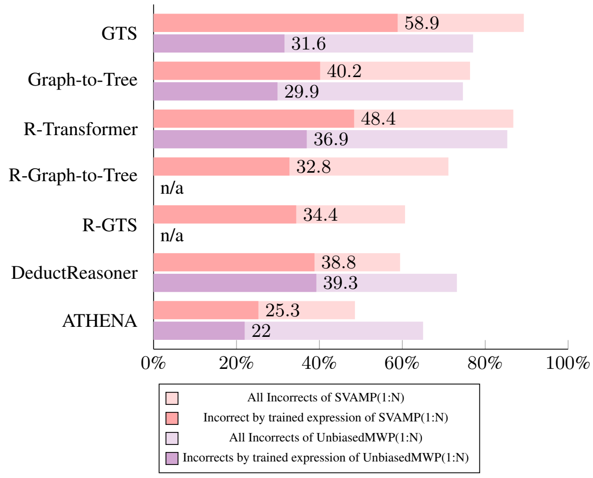

Figure 4: The percentage the incorrect answers match with the answers to different questions that share the context from the training set. The less the percentage scores, the less the method unnecessarily leans on the training bias.

but ask for different target answers. We observe from Tabel 3 that the results show ATHENA is strong at applying mathematical reasoning that is formed by unlearned patterns once the model has learned the context. Our approach is distinguished from other baselines including RoBERTa-GTS and DeductReasoner which show the opposite phenomena. Other baselines are relatively stronger on original benchmarks than on the benchmark variants including those with the one-to-many Test. Hence we reach the conclusion that ATHENA has the superiority of acknowledging the subtlety of contextual information governed by the required mathematical operations.

Dependence on Training Set. We observe that ATHENA performs well on datasets that apply the one-to-many test because our model has a sense of subtlety in terms of distinct question concepts, not because our model is reluctant to follow learned expressions. Figure 4 illustrates where the wrong prediction for the question variant experiments comes from. If a model outputs equations that are labeled for questions with shared contexts when being trained, this indicates that the model relies on training data points, especially on context contents regardless of different question expressions. The result shows that our model also has the least accuracy for a golden training example. It is notable that ATHENA has the least score for following the trained expressions while DeductiveReasoner predicts the highest scores among other baselines that use RobBERTa, even higher than those of R-GTS or R-Graph-to-Tree on UnbiasedSVAMP(1:N). This shows that while DeudctiveReasoner can learn to solve mathematical problems, it also easily falls into learning shortcuts.

Different Sizes of PLMs. We estimate the baselines both on RoBERTa-base and RoBERTa-large models to examine the influence of the model sizes. As expected, Table 2 shows that the bigger the model size is for the embedding, the better the model performance reaches. When we estimate the accuracy gaps by increasing the model size, ATHENA achieves relatively better performance gains (7.26%p) on average for the entire benchmarks than DeductReasoner does (4.6%p). We can observe that on dataset variants, ATHENA obtains relatively more benefits from bigger model sizes (14.15%) than DeductReasoner does (8.15%p), while both are still taking great advantage of the rich model parameters to understand the question better and to solve those confusing questions. It also shows that DeductReasoner fails to improve performance on question variants from the original datasets leveraging the additional training sets in large-scale PLM. In short, our model leverages large-scale PLM much more efficiently than the competitive model.

Visualization of Thoughts. We interpret the thoughts using attention scores between reasonable thoughts and the problem sequence. [6] As illustrated in Figure 5, we observe how the thought relates to the words. Most of the initial thoughts are related to the "playground", while the thoughts carrying the meaning of increased size show a strong

[6] We use answer layer to calculate the attention score, giving the problem sequence embedding as an input, instead of the goal vector.

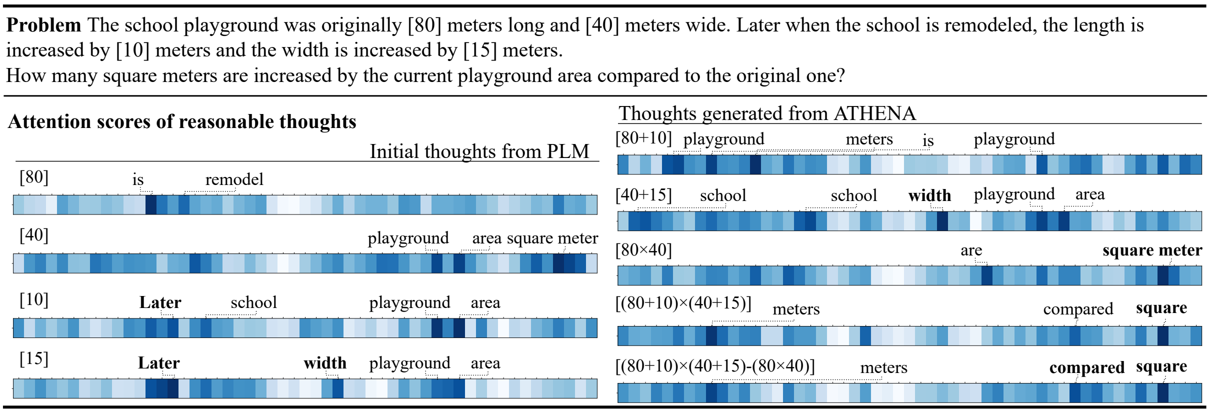

Figure 5: Visualization of reasonable thoughts from ATHENA with calculating attention score of the tokens in the problem sequence on RoBERTa-large.

	MAWPS	ASDiv-A	SVAMP	Math23k	UnbiasedMWP	SVAMP (1:N)	UnbiasedMWP (1:N)	Average
Avg depth	3.87	3.46	3.47	5.18	4.44	3.47	4.44	4.05
ATHENA	**92.2**	**86.4**	**45.6**	**85.1**	36.2	**52.5**	**35.4**	**62.0**
− update	92.1	84.8	44.9	82.7	34.9	52.4	34.7	60.9
−(reason+update)	90.6	85.0	44.7	65.7	**36.3**	51.5	34.6	58.3

Table 3: Ablation studies on reasonable thought mechanism

correlation to the word "Later". The thoughts carrying width sizes [15] and [40+15] show high attention scores on "width", while the other thoughts do not have high attention scores on them. Thoughts that calculate the area produce high attention scores on words "square meter" or "area". The final thought marks a high score on "compared", which asks for the difference between the increased and original areas.

Ablation on Reasonable Thought. We conduct an ablation study to evaluate how ATHENA composes the reasonable thought mechanism to ultimately generate optimal final thoughts. For evaluating the impact of component functions in generating reasonable thoughts, we adopt two different settings:

(1) we do not *update* reasoning vectors but use the initial reasoning vector (i.e., [CLS] token) in all expansion depths $\mathrm{R}_d = \mathrm{R}_0$. We aim to see how the existence of thoughts that update the reasoning vector impacts models to help find solid reasonable thoughts. (2) we do not even start *reason*ing in the reasonable thought procedure and directly classify whether thoughts are usable for the next iteration: $\text{use}(\theta) = \sigma(\theta W_r + b_r)$.

Table 3 shows that ATHENA takes full advantage of the reasonable thought stage via reasoning with the reasoning vector and their updating strategy. Despite slight fluctuations across different methods, ATHENA without reasoning function decreases the overall performances by up to 3.7%p compared to our proposed ATHENA. When ATHENA does not update the reasoning vectors in the thought expansion iteration while still adopting the reasoning function, the performance decreases relatively by 1.1%p. From those observations, we conclude that the decent performance of ATHENA comes from a grounded reasoning vector refined by reasoning and updating strategies.

5 Conclusion

We state that an ideal MWP method needs to be practical in real-world settings that are critical to capture the lexical sophistication of the same mathematical operations. For this reason, we conclude that ATHENA with thought expansion reaches significant improvements toward the ideal WMP method due to its decent performance on unseen problems or restricted examples to learn.

Limitations

This work only considers arithmetic problems, not algebraic, calculus, or other topics of mathematical problems. As many other works do, we only consider math word problem datasets having single equations and we do not verify with any dataset for multiple equations.

References

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Zhenya Huang, Xin Lin, Hao Wang, Qi Liu, Enhong Chen, Jianhui Ma, Yu Su, and Wei Tong. 2021. Disenqnet: Disentangled representation learning for educational questions. KDD '21, page 696–704, New York, NY, USA. Association for Computing Machinery.

Zhenya Huang, Qi Liu, Weibo Gao, Jinze Wu, Yu Yin, Hao Wang, and Enhong Chen. 2020. Neural mathematical solver with enhanced formula structure. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1729–1732, New York, NY, USA. Association for Computing Machinery.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5944–5955, Dublin, Ireland. Association for Computational Linguistics.

Philip Johnson-Laird. 2008. *How we reason*. Oxford University Press.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 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*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):13188–13190.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 997–1009, Seattle, United States. Association for Computational Linguistics.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4232–4240.

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

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing English math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, Online. Association for Computational Linguistics.

Arindam Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, Berlin, Germany. Association for Computational Linguistics.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

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 the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1473–1482, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. 2020. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10524–10533. PMLR.

Zhicheng Yang, Jinghui Qin, Jiaqi Chen, and Xiaodan Liang. 2022. Unbiased math word problems benchmark for mitigating solving bias. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1401–1408, Seattle, United States. Association for Computational Linguistics.

Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3384–3394, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937, Online. Association for Computational Linguistics.

Appendices

A Training Details

In this section, we provide detailed information about our training settings.

Loss. Given an answer equation $\mathcal{E}$, let $\mathbb{1}_{\text{reason}}(\theta)$ denote the target of reason for a thought θ and $\mathbb{1}_{\text{answer}}(\theta)$ denote the target of the final decision answer for a thought θ:

$$\mathbb{1}_{\text{reason}}(\theta) = \mathbb{1}(\mathcal{E}(\theta) \subset \mathcal{E}), \mathbb{1}_{\text{answer}}(\theta) = \mathbb{1}(\mathcal{E}(\theta) \equiv \mathcal{E}),$$

where $\mathcal{E}(\theta) \subset \mathcal{E}$ denotes that $\mathcal{E}$ contains the sub-expression $\mathcal{E}(\theta)$ (e.g., $(a+b) \subset (a+b) \times c$).

Let BCE denote the binary cross entropy function, the training objective is to minimize the loss $\mathcal{L}$:

$$\mathcal{L} = \frac{\sum\limits_{\theta \in \bigcup_d \Theta_d} BCE(\text{reason}(\theta), \mathbb{1}_{\text{reason}}) + \sum\limits_{\theta \in \Theta_d^*} BCE(\text{answer}(\theta), \mathbb{1}_{\text{answer}})}{|\bigcup_d \Theta_d| + \Theta_d^*}.$$

Optimizer. We use AdamW optimizer (Loshchilov and Hutter, 2017) with weight decay $\omega = 1e-5$. Learning rate lr_e for each epoch e is decayed every S_{lr} epoch with factor γ starting from lr:

$$lr_e = lr \cdot \gamma^{\lfloor e/S_{lr} \rfloor}.$$

Regularization. We adopt dropout with probability p to every layer and stochastic weight averaging (Izmailov et al., 2018) for last $epoch_{swa}$ epochs.

Hyperparameters. We present our experiments for hyperparameters in Table 4, with the bold text denoting the best performance. We train our model for 100 epochs. In the result, we observe that RoBERTa-base and RoBERTa-large share the best hyperparameter settings except for the learning rate lr.

	Batch Size	lr	S_{lr}	γ	p	$epoch_{swa}$
RoBERTa-base	[**4**, 8]	[5e-6, 7e-6, 1e-5, **1.3e-5**, 1.5e-5, 2e-5]	[**10**, 15, 20]	[**0.5**, 0.7]	[0.1, **0.5**]	[**30**, 50, 70]
RoBERTa-large	[**4**, 8]	[5e-6, **7e-6**, 1e-5, 1.3e-5, 1.5e-5, 2e-5]	[**10**, 15, 20]	[**0.5**, 0.7]	[0.1, **0.5**]	[**30**, 50, 70]

Table 4: Hyperparameter search spaces of ATHENA

B Statistics of One-to-Many Split

In Section 4.1, we explain building one-to-many dataset splits. We provide how many groups and examples are made from the contexts in Table 5.

	SVAMP (1:N)	**UnbiasedMWP (1:N)**
# examples in original split	3138 / 0 / 1000	2507 / 200 / 685
# groups of single examples	438	45
# groups of multiple examples	205	154
# examples in one-to-many split	3343 (+205) / 438 (+438) / 357 (-562)	2661 (+154) / 245 (+45) / 486 (-199)

Table 5: Statistics of one-to-many test splits

C Statistics of Thoughts

In this section, we show various statistics of thoughts and reasoning that each dataset requires. While Math23k requires a large number of candidate thoughts in total depth, we show a thought expansion in each depth does not require huge memory space. Therefore, efficient implementation strategies such as removing unselected candidate thoughts from memory space are enough to manage computational resources.

Dataset	# candidates in total depth			# in a reasoning path			# candidates in last depth			depth of reasoning path		
	min	average	max	min	average	max	min	average	max	min	average	max
MAWPS	17	45.40±0.46	192	2	4.52±0.03	12	4	9.49±0.08	48	2	3.87±0.03	11
ASDiv-A	16	26.86±0.42	71	3	4.10±0.03	7	6	9.65±0.09	22	1	3.46±0.02	5
SVAMP	2	28.09±0.44	70	1	4.23±0.03	7	2	10.54±0.10	22	1	3.47±0.03	5
Math23k	4	65.1±0.31	939	1	6.33±0.02	29	2	14.85±0.06	108	1	5.18±0.01	41
U.MWP	5	47.0±0.47	214	1	5.18±0.03	13	2	11.67±0.11	48	1	4.44±0.02	11

Table 6: Statistics of thoughts that are required for each dataset

D Effect of Punctuation Mark

In Section 3, we initialize goal vector G with the punctuation mark of the question sequence or the last punctuation mark (i.e., the question mark in most cases). The motivation of this strategy is from Clark et al. (2019) showing the punctuation mark gets high attention from other tokens in the last layers. Intuitively, high attention can generalize the question sequence, so we conduct experiments to evaluate the generalization ability of the punctuation mark compared to using all question sequences as a goal vector G. We conduct experiments for all datasets except Math23k (Wang et al., 2017) since it does not provide the explicit question sequence annotation.

As shown in Table 7, using the punctuation mark effectively generalize the question to represent a goal in most cases. It shows even better performances than using the question sequence. Intuitively the question sequence holds some tokens that are not informative for reasoning, so generalizing with the punctuation mark helps the model to focus on a goal of reasoning.

	MAWPS	ASDiv-A	SVAMP	UnbiasedMWP	SVAMP (1:N)	UnbiasedMWP (1:N)	Average
Avg depth	3.87	3.46	3.47	4.44	3.47	4.44	4.05
RoBERTa-base							
punctuation mark	**92.2**	**86.4**	**45.6**	36.2	**52.5**	**35.4**	**58.1**
question sequence	92.0	86.3	44.9	**36.3**	51.0	33.4	57.3
RoBERTa-large							
punctuation mark	**93.0**	91.0	**54.8**	**42.0**	**67.8**	**48.4**	**66.2**
question sequence	92.9	**91.2**	54.4	41.0	66.9	46.8	65.5

Table 7: Comparing goal vector using the whole question sequence from the punctuation mark