Supplementary Materials for
Learning by Fixing: Solving Math Word Problems with Weak Supervision

Anonymous Submission ID 5790

Goal Decomposition in GTS

Given the goal $\mathbf{q}$ and the context $\mathbf{c}$ of a node, the two sub-goals $\mathbf{q}_l, \mathbf{q}_r$ are generated by a feedforward neural network with gating mechanism:

$$\begin{aligned}\mathbf{h}_l &= \sigma(\mathbf{W}_{ol}[\mathbf{q},\mathbf{c},\mathbf{e}(\hat{t})]) \odot \tanh(\mathbf{W}_{cl}[\mathbf{q},\mathbf{c},\mathbf{e}(\hat{t})]) \\ \mathbf{q}_l &= \sigma(\mathbf{W}_{gl}\mathbf{h}_l) \odot \tanh(\mathbf{W}_{le}\ \mathbf{h}_l)\end{aligned} \tag{1}$$

$$\begin{aligned}\mathbf{h}_r &= \sigma(\mathbf{W}_{or}[\mathbf{q},\mathbf{c},\mathbf{e}(t)]) \odot \tanh(\mathbf{W}_{cr}[\mathbf{q},\mathbf{c},\mathbf{e}(t)]) \\ \mathbf{q}_r &= \sigma(\mathbf{W}_{gr}[\mathbf{h}_r,\mathbf{e}_l]) \odot \tanh(\mathbf{W}_{re}[\mathbf{h}_r,\mathbf{e}_l])\end{aligned} \tag{2}$$

Since the tree is generated with preorder traversal, the generation of the right sub-goal takes into account the left-child tree. It is achieved by injecting the left-child tree's embedding $\mathbf{e}_l$ into Equation 2.

Tree Embedding. We adopt a recursive neural network to encode the tree. For a tree with a root node $\hat{t}$, its embedding is defined recursively as:

$$\mathbf{e} = \begin{cases}\mathbf{e}(\hat{t}), & \text{if } \hat{t} \in V^{num} \cup V^{con} \\ \text{comb}(\mathbf{e}_l,\mathbf{e}_r,\hat{t}), & \text{if } \hat{t} \in V^{op}\end{cases} \tag{3}$$

$$g_t = \sigma(\mathbf{W}_{gt}[\mathbf{e}_l,\mathbf{e}_r,\mathbf{e}(\hat{t})]) \tag{4}$$

$$C_t = \tanh(\mathbf{W}_{ct}[\mathbf{e}_l,\mathbf{e}_r,\mathbf{e}(\hat{t})]) \tag{5}$$

$$\text{comb}(\mathbf{e}_l,\mathbf{e}_r,\hat{t}) = g_t \odot C_t \tag{6}$$

where $\mathbf{e}_l, \mathbf{e}_r$ are the embeddings of the left-child tree and the right-child tree. The recursion stops when reaching the leaf nodes of this tree.

Learning by REINFORCE

Since the execution of the generated solution tree is symbolic and non-differentiable, it is infeasible to use back-propagation to compute gradients. A straightforward approach is to employ policy gradient methods like REINFORCE (Williams 1992) to train the neural model. Specifically, we regard the GTS model $p_\theta(T|P)$ as the policy function and the reward can be computed as:

$$r(T,y) = \begin{cases} 0, & \text{if } \text{exec}(T) \neq y \\ 1, & \text{if } \text{exec}(T) = y\end{cases} \tag{7}$$

where $\text{exec}(T)$ denotes the result of executing T. With the REINFORCE algorithm, we compute the gradients as:

$$\begin{aligned}\nabla_\theta &= r(T,y)\nabla_\theta \log p_\theta(T|P), T \sim p_\theta(T|P) \\ &= \begin{cases} 0, & \text{if } \text{exec}(T) \neq y \\ \nabla_\theta \log p_\theta(T|P), & \text{if } \text{exec}(T) = y\end{cases}\end{aligned} \tag{8}$$

Equation 8 reveals the gradient is non-zero only when the sampled solution tree T can generate the ground-truth answer y. However, among the whole space of T, only a very small portion can generate the desired y, which implies that the REINFORCE will get zero gradients from most of the samples. This is the main reason that the REINFORCE method converges slowly or even fails to converge, as also shown in the experiments.

Training Details

The models are implemented using PyTorch on an Ubuntu system with an Nvidia GTX2080Ti GPU. For the GTS model, the word embedding size is 128, and the bi-directional GRU has 512 hidden neurons. The learning rate is initialized as 0.001 and gets halved every 20 epochs. The batch size is 64 for both exploring and learning stages. We set the dropout rate (Hinton et al. 2012) to 0.5 and weight decay to $1e^{-5}$. All models are trained with 200 epochs using Adam optimizer (Kingma and Ba 2015).

Qualitative Examples for 1-FIX

Figure 1 shows an illustrative example of the 1-FIX process implemented with a priority queue. The visiting priority for each fix-tuple in the priority queue is defined as following:

$$p(A \rightarrow \alpha_A) = \begin{cases} \frac{1-p(A)}{p(A)}, & \text{if } A \notin \Sigma \\ \frac{p(\alpha_A)}{p(A)}, & \text{if } A \in \Sigma \ \&\ \alpha_A \in \Sigma,\end{cases} \tag{9}$$

where Σ is the set of possible leaf nodes in the reasoning tree, *i.e.*, the target vocabulary defined in the GTS model. If $A \in \Sigma$, $p(A)$ is calculated as Equation 4 in the main paper; otherwise, it is defined as the product of the probabilities of all leaf nodes in A. If $A \in \Sigma$ and $\alpha_A \notin \Sigma$, it means we need to correct a leaf node to a token that is not in the vocabulary. Therefore, this fix is impossible and then discarded.

Figure 2 shows several illustrative examples of correcting the wrong solutions using the 1-FIX algorithm.

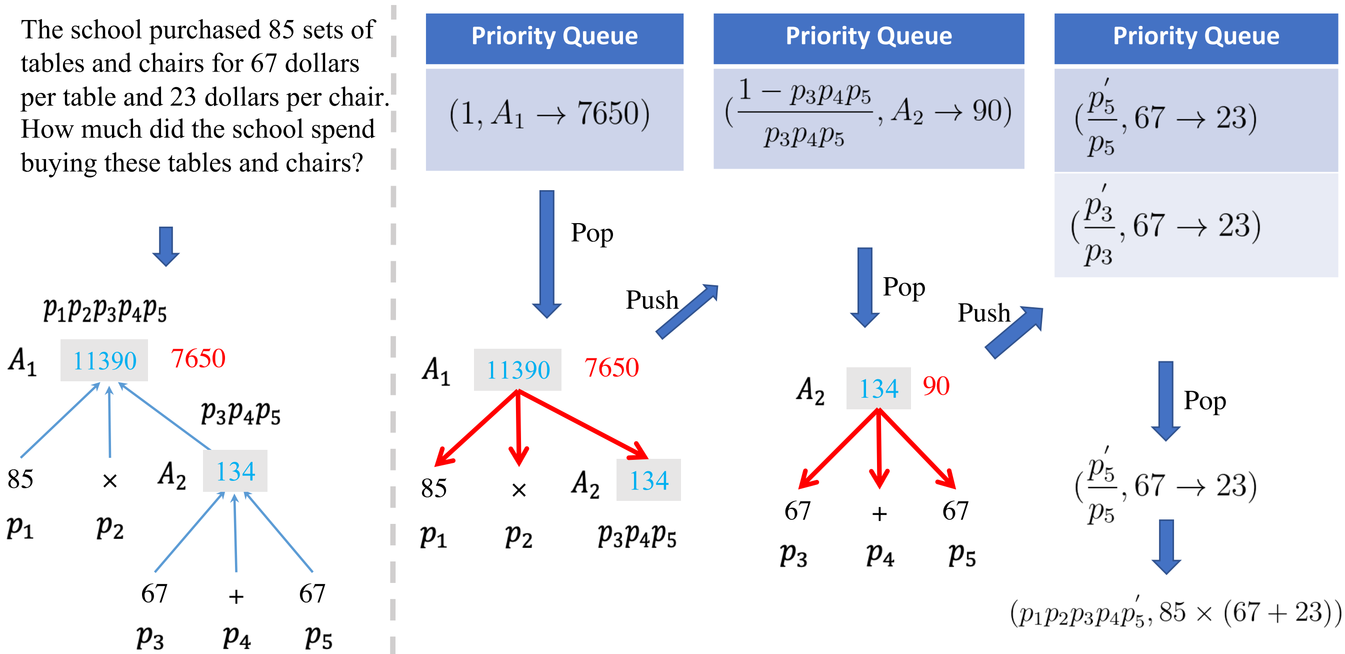


Figure 1: An illustrative example of the 1-FIX algorithm with the priority queue.

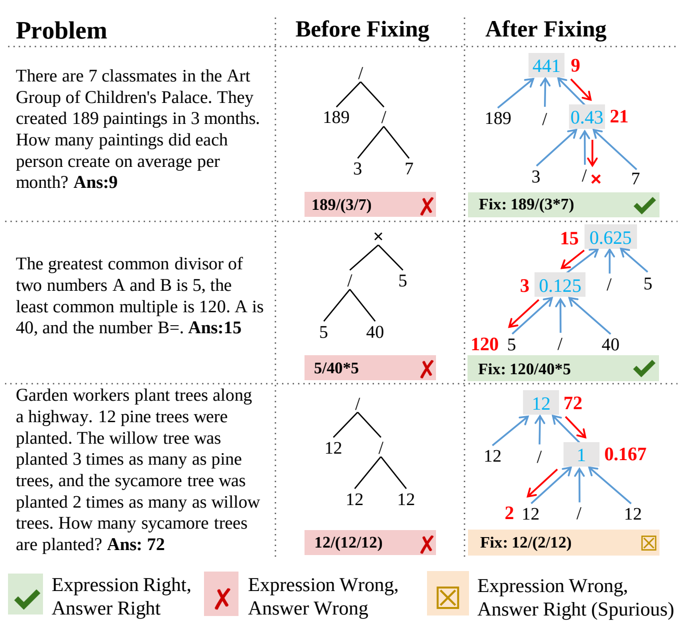


Figure 2: Illustrative examples of correcting the wrong expressions using the 1-FIX algorithm.

Additional Qualitative Examples

To analyze our model's ability to generate multiple reasonable solutions for a single problem, we visualize more examples of the top-5 predictions of GTS-LBF, as shown in Figure 3.

For the first and second example in Figure 3, our method manages to generate diverse solutions for a given problem. The first problem can be solved by calculating the remaining ratio and then multiplying the ratio by the total length. Another solution is to subtract the length used from the total length. The second problem can be solved by summing up the number of willow trees and pine trees. The number of willow trees can be"38×2" or "$38 + 38$", which are all generated by our model. Alternatively, it can be solved by plus the triple the number of willow trees with ten.

Example 3-5 are failure cases of our method. For Example 3, the error comes from the wrong prediction of the problem goal. Our method also generates spurious expressions. The fifth expression of Example 3 is a spurious solution because of redundant symbols with no practical meaning. The first expression of Example 4 is also a spurious solution because it does not have prior knowledge about how to calculate the volume of a cylinder and hits the right answer accidentally based on wrong guesses. Our method fails to generate the right expression for the fifth example because it cannot figure out the exact relation between vegetables.

Visualization of Attention Map

To further analyze why wrong examples are generated, we visualize the attention of the GTS model. Examples are shown from Figure 4 to Figure 7.

In Figure 4, when the model generates the root goal, it pays attention to words like "compare" and "more" and generates a "+", then it pays attention to "1200 books" and "actual purchase," therefore generating the subgoal of actual purchase books. To decompose this goal, our model first pays attention to "1200" and then "20%", so that the number of purchase books is calculated. Note that the model pays attention to the last sentence when generating the numbers, and we can hypothesize that it uses this attention to remind itself the goal is to measure the actual purchase books. Fig-

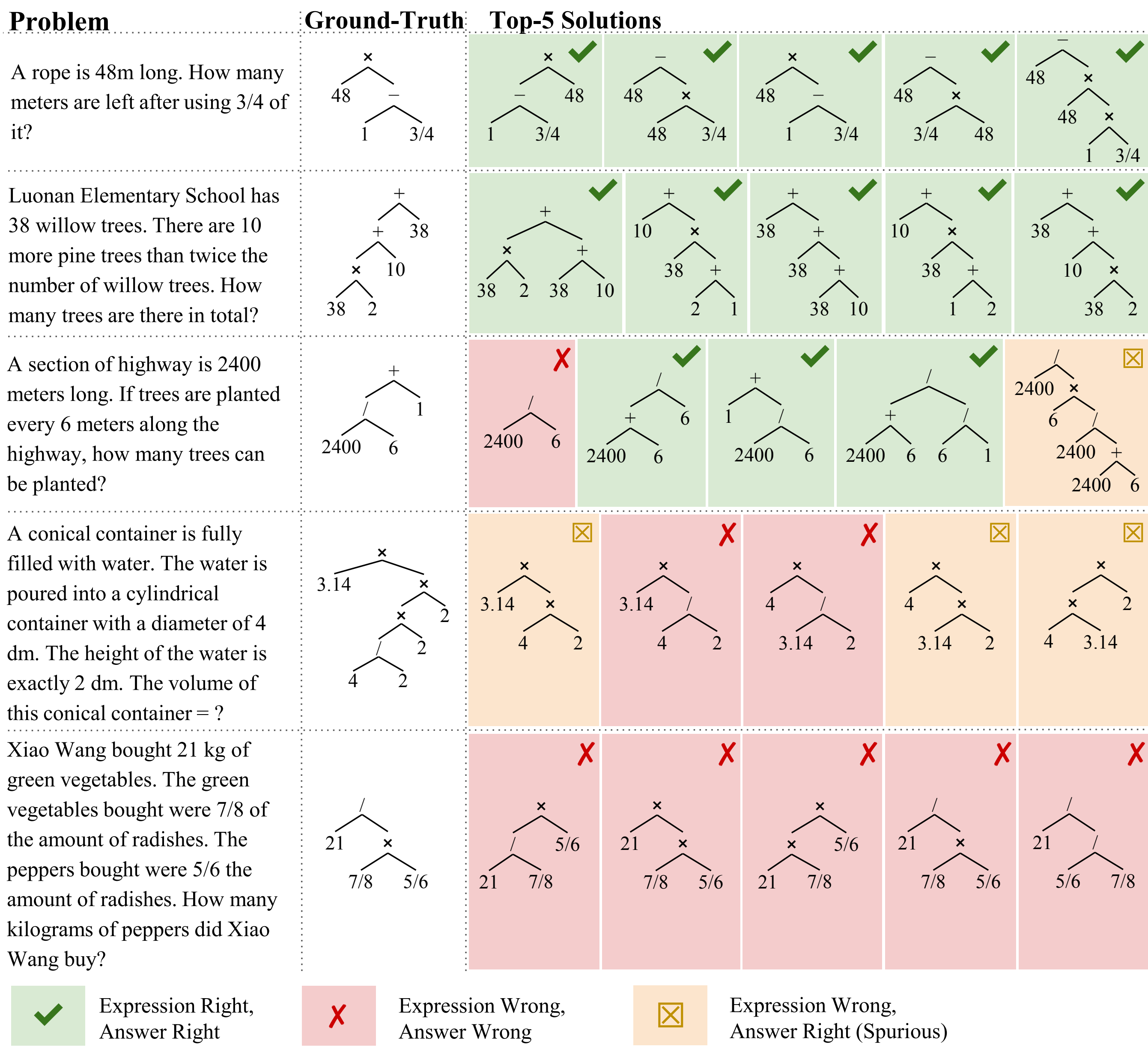


Figure 3: Qualitative Results on the Math23K dataset. We visualize the solution trees generated by our method.

ure 5 shows another example. Our model first pays attention to the last sentence "days finish task" and knows its goal is to calculate the days to complete the task. It then pays attention to "more" and generates token "+".

We further analyze some failure examples. In Figure 6, the goal should be the rate. However, the model first pays attention to the "saplings" and "death rate" instead of the survival rate. Therefore, it mistakenly generates the goal of calculating the dead saplings. Figure 7 makes a mistake in determining the relation. It pays the correct attention to "flour compare rice more". However, it understands it wrong and calculates "how many more kilograms of rice are there than flour" instead of "how many more kilograms of flour". Therefore, in the next step, it pays attention to words like "kilometers", "6", "bags", and calculates rice's weight first.

Problem	Model	Buffer
A truck travels 100 km in 2 hours. At this speed, if it travels for another 3.5 hours, how many kilometers will it complete for the entire journey?	GT	$100/2 * (2 + 3.5)$
	LBF	$100/(2/(2 + 3.5))$, $100 + (100/(2/3.5))$, $100/(2/(3.5 + 2))$, $100/2 * 3.5 + 100$, $100 + 100 * 3.5/2$
Zhang ran in a 1000-meter running race, and ran 320 meters per minute for 3 minutes. How many meters are there from the finish line?	GT	$1000 - 320 * 3$
	LBF	$1000 - 3 * 320$, $1000 - 320 * 3$, $1000 - 320 - 320 - 320$, $1000 - 320 * 3/1$
The price of a calculator is 35 dollars The price of a computer is 90 times of it, if we buy a computer and a calculator, how many dollars do we pay?	GT	$35 * 90 + 35$
	LBF	$35 * 90 + 35$, $(1 + 90) * 35$, $35 + 35 * 90$, $35 * (1 + 90)$, $35 * (90 + 1)$

Table 1: Examples of memory buffer compared with the ground truth expression given by Math23K.

Problem	Model	Predictions
The road between two places is 342 kilometers long. A and B are starting from two places on motorcycles. How many hour later do they meet?	GTS	$342/(47.5 + 47.5 * 4/5)$
	GTS-LBF-w/o-M	$342/(47.5 + 47.5 * 4/5)$, $(342/(47.5 + 4/5 * 47.5)$
	GTS-LBF	$342/(47.5 * (4/5 + 1)))$, $(342/(47.5 + 47.5 * 4/5)$, $(342/(47.5 * (1 + 4/5))$
	PT-GTS-LBF	$342/(47.5 + (47.5 * (4/5)))$, $(342/(47.5 * (1 + (4/5)))$, $(342/(47.5 + ((4/5) * 47.5))$, $(342/(47.5 * ((4/5) + 1))$
The school purchased 85 sets of tables and chairs for 67 dollars per table and 23 dollars per chair. How much did school spend buying these tables and chairs?	GTS	$(67 + 23) * 85, 85 * (67 + 23)$
	GTS-LBF-w/o-M	$85 * (67 + 23), (67 + 23) * 85$, $85 * (23 + 67)$
	GTS-LBF	$85 * (67 + 23), 67 * 85 + 23 * 85$, $(67 + 23) * 85, 85 * (23 + 67)$,
	PT-GTS-LBF	$85 * (67 + 23), 67 * 85 + 23 * 85$, $(67 + 23) * 85, 85 * (23 + 67)$, $(23 + 67) * 85$
Ming read 15 pages of books in 5 days. According to this, how many pages can he read in a month (30 days)?	GTS	$15/5 * 30$
	GTS-LBF-w/o-M	$15/(5/30), 15 * 30/5$
	GTS-LBF	$15/5 * 30, 30 * 15/5$, $15 * 30/5, 15/(5/30)$
	GTS-LBF-pretrain	$15/5 * 30, 15 * 30/5$, $15/(5/30)$, $15 * 30/5, 30 * 15/5$

Table 2: Beam search results of different methods.

References

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv* abs/1207.0580.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4): 229–256.

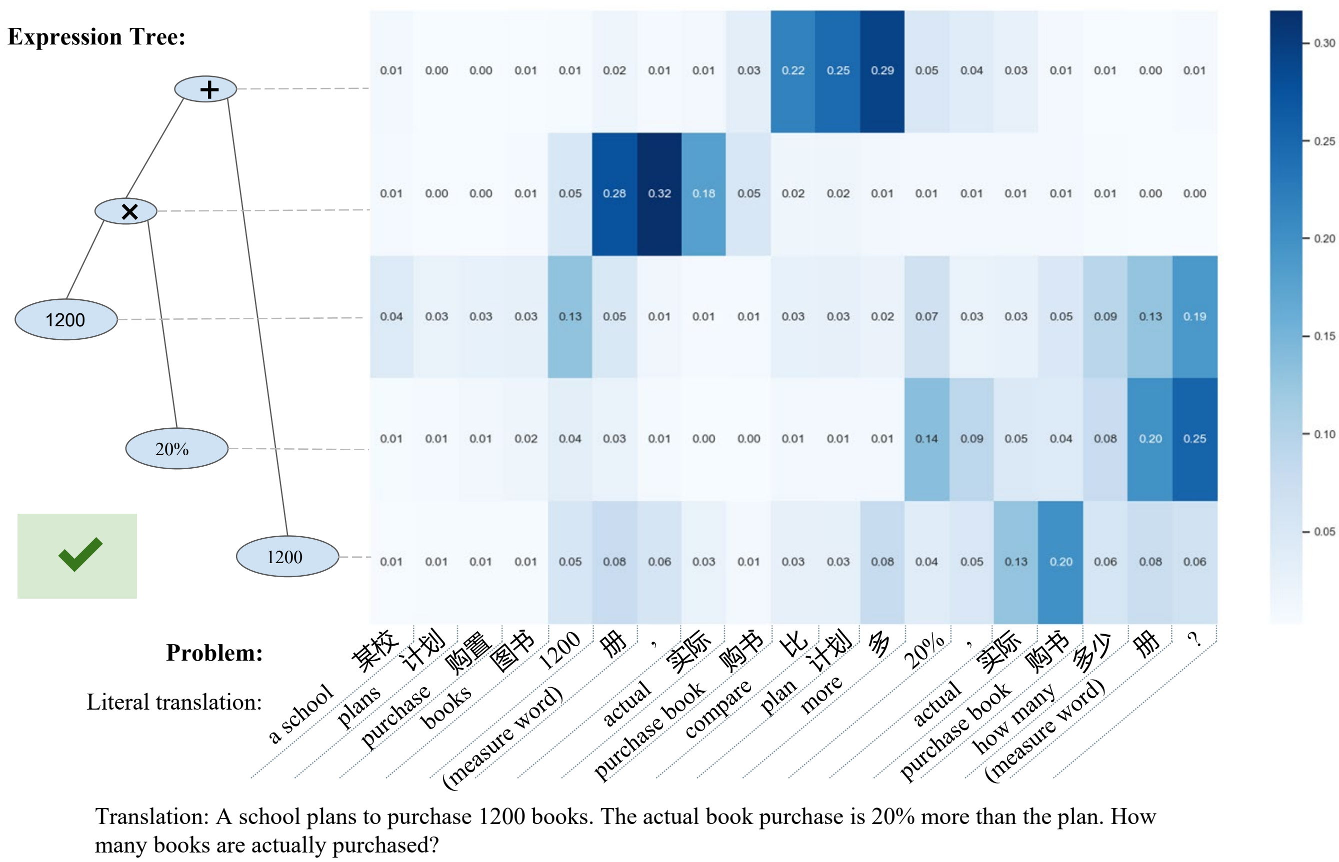
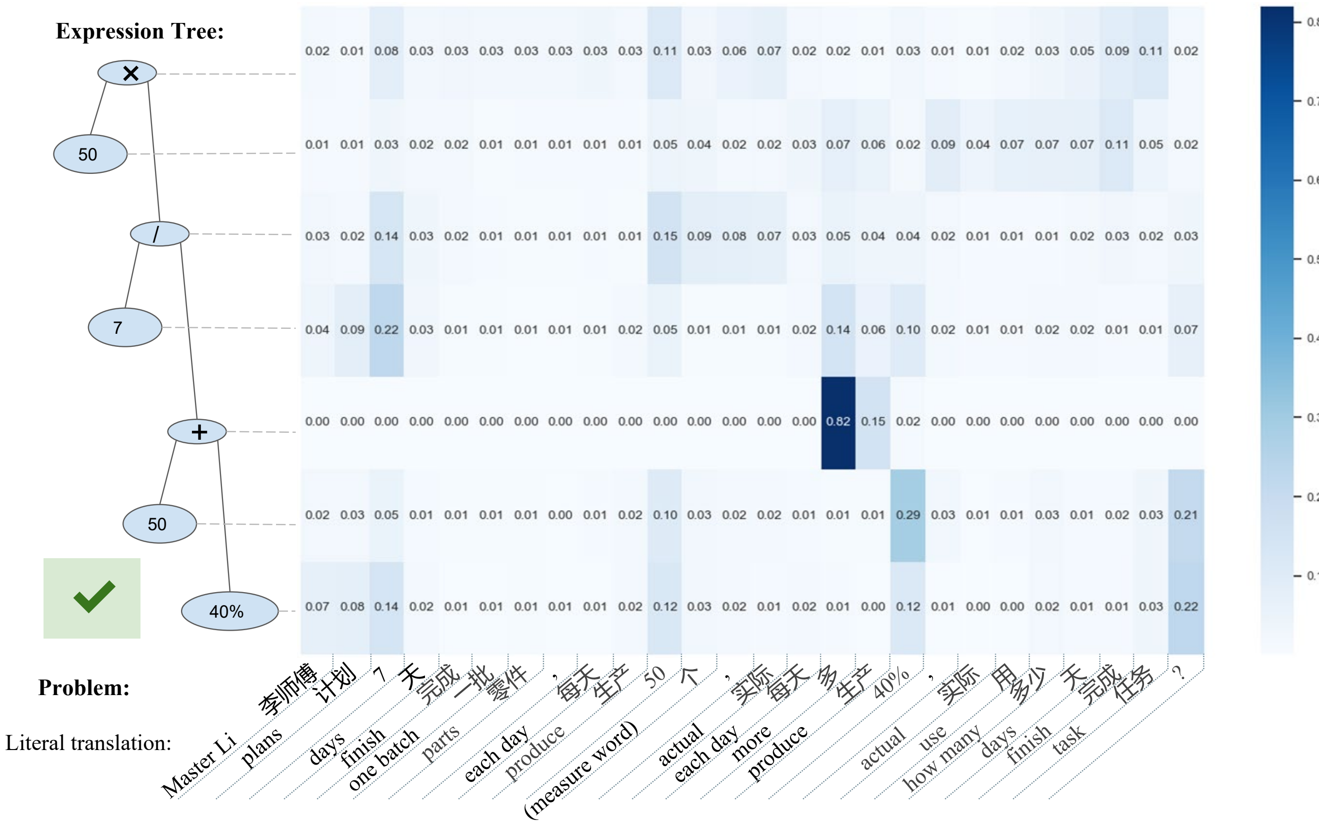
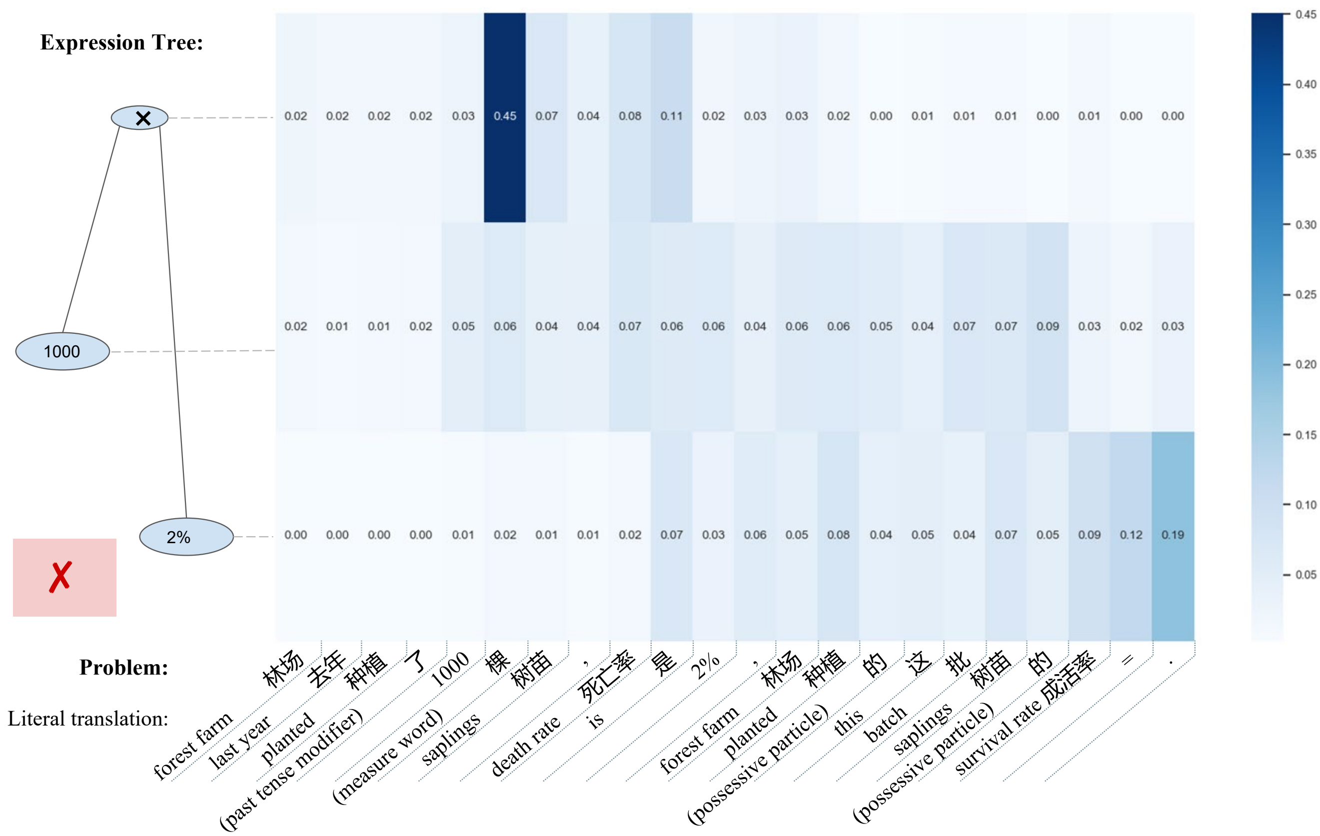


Figure 4: Attention map of a successful example.



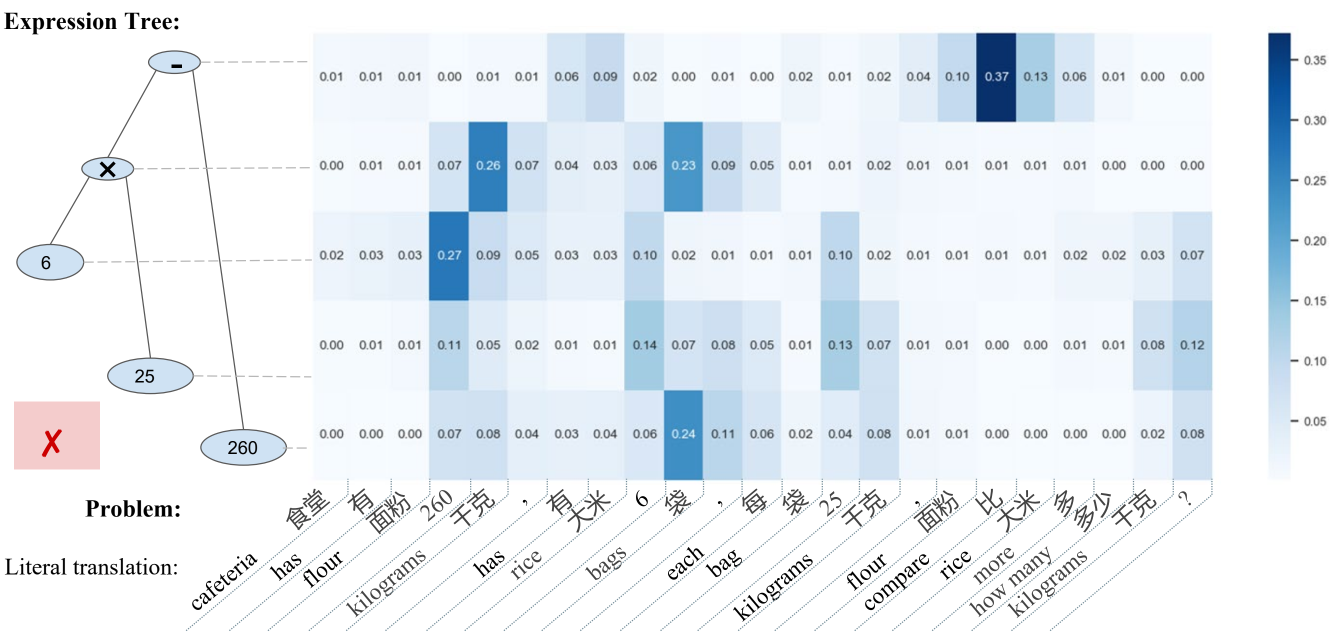
Translation: Master Li plans to complete a batch of parts in 7 days, producing 50 pieces per day. He actually produces 40% more per day than his plan. How many days does it actually take to complete the task?

Figure 5: Attention map of a successful example



Translation: The forest farm planted 1,000 saplings last year, and the mortality rate was 2%. The survival rate of the saplings planted in the forest farm = ?

Figure 6: Attention map of a failed example



Translation: The cafeteria has 260kg of flour and 6 bags of rice, 25kg per bag. How many more kilograms of flour are there than rice?

Figure 7: Attention map of a failed example