EPT-X: An Expression-Pointer Transformer model that generates eXplanations for numbers

Bugeun Kim **Kyung Seo Ki** **Sangkyu Rhim** **Gahgene Gweon**
Department of Intelligence and Information, Seoul National University
Seoul, Republic of Korea
{cd4209, kskee88, sk.rhim, ggweon}@snu.ac.kr

Abstract

In this paper, we propose a neural model EPT-X (Expression-Pointer Transformer with Explanations), which utilizes natural language explanations to solve an algebraic word problem. To enhance the explainability of the encoding process of a neural model, EPT-X adopts the concepts of plausibility and faithfulness which are drawn from math word problem solving strategies by humans. A plausible explanation is one that includes contextual information for the numbers and variables that appear in a given math word problem. A faithful explanation is one that accurately represents the reasoning process behind the model's solution equation. The EPT-X model yields an average baseline performance of 69.59% on our PEN dataset and produces explanations with quality that is comparable to human output. The contribution of this work is two-fold. (1) EPT-X model: An explainable neural model that sets a baseline for algebraic word problem solving task, in terms of model's correctness, plausibility, and faithfulness. (2) New dataset: We release a novel dataset PEN (Problems with Explanations for Numbers), which expands the existing datasets by attaching explanations to each number/variable.

1 Introduction

Algebraic word problem solving is a challenging task for understanding natural language. As shown in Table 1, a model needs to interpret a word problem into a solution equation to solve the problem. Recent neural approaches have employed encoder-decoder architecture to tackle this task and achieved remarkable answer correctness (Huang et al., 2018; Chiang and Chen, 2019; Amini et al., 2019; Kim et al., 2020; Ki et al., 2020): ranging from 65% to 84% depending on datasets. So, as the model delivers a plausible answer, researchers have a firm belief that an encoder component of a neural model can comprehend the problem correctly. However, this belief has less been verified due to the opaqueness of an encoder. Thus, this paper aims to design an architecture that reveals how it understands a given word problem.

Q. Tom has 12 coins in quarters and nickels. Their value is $2.20. How many coins of each type does he have?	
Solution Equation	$x + y = 12,\ 0.25x + 0.05y = 2.20$ $(\because x = 8, y = 4)$
Explanation	12 = the total number of coins 2.20 = the total value of coins x = the number of quarters y = the number of nickels

Table 1: An example algebraic word problem

Our novel model, Expression-Pointer Transformer with Explanations (EPT-X), is inspired by some pedagogical studies about human strategies on understanding an algebraic word problem (Conway and Polya, 1985; Jitendra et al., 2007; Montague, 2008; Jitendra and Star, 2012). In classrooms, teachers ask students to make an explanation or a diagram that depicts the role of each number written in the problem. Then, students use these explanations for numbers to build a correct equation. That is, understanding a problem produces explanations that satisfy the following two criteria. (1) *plausibility*: A plausible explanation is one that includes contextual information for the numbers and variables that appear in a given math word problem. Especially, as humans recognize each number/variable individually, the explanations should reveal what each number or variable represents in the context of the given problem. (2) *faithfulness*: A faithful explanation is one that accurately represents the reasoning process behind the model's solution equation (Jacovi and Goldberg, 2020). In other words, they should imply a reason behind selecting operators or operands.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 4442 - 4458

To reflect these two criteria in EPT-X, we adopt a two-phase architecture: (1) explaining numbers/variables and (2) building solution equations.

Though Ling et al. (2017) attempted to generate explanations, this work is different from ours in that their model focused on explaining the decoding process. So, they have less explored the above two criteria. In contrast, this paper attempts to explain how the model understands the given word problem by modifying an encoder component of a neural model. As humans successfully solve word problems by explaining their understanding, we expect our EPT-X model to achieve a good performance in terms of three criteria: *correctness* of equations, *plausibility* of explanations, and *faithfulness* of explanations. Through several analyses, our paper shows the following two contributions:

1. *EPT-X model:* We propose a baseline model that can generate explanations and solve algebraic word problems, in terms of correctness, plausibility, and faithfulness.
2. *New dataset:* We release a novel dataset PEN (Problems with Explanations for Numbers), which expands the existing datasets by attaching explanations to each number/variable.

2 Related work

Correctness: Researchers have attempted to build a model that solves word problems. Early attempts used hand-crafted features collected by experts to make a model understand a word problem (Kushman et al., 2014; Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Zhou et al., 2015; Upadhyay et al., 2016; Roy and Roth, 2017). Although researchers can interpret these models using the features, extending these studies to other datasets is limited as designing features is labor-intensive. On the other hand, recent studies have employed neural models (Wang et al., 2017; Huang et al., 2018; Chiang and Chen, 2019; Amini et al., 2019; Kim et al., 2020; Ki et al., 2020) and achieved answer correctness ranging from 62% to 84%. Though their extensibility is better than hand-crafted features, it becomes harder to interpret how a neural model understands a word problem.

Plausibility: To make a neural model that explains its reasoning process, Ling et al. (2017) built a model that outputs both a computation process and a rationale behind the process. Though their model generated a natural language phrase that explains a computation step in advance, the model is not enough to meet the plausibility criterion because of two issues. First, it is not guaranteed whether their model explains all numbers and variables required to solve the problem. As they focused more on explaining the model's computation, their model often skips explaining its understanding of numbers and variables stated in a problem. Second, it is not confirmed whether their model generates rationale comparable to that of humans. Though they measured their quality of rationale using BLEU-4 (Papineni et al., 2002), the reported score of 27.2 is somewhat low and not compared with any human-level performance. We suspect that this low-quality explanation affected the low correctness of their model: 36.4%. Therefore, it is worthwhile to build a new model that fulfills the plausibility criterion.

Faithfulness: As far as we know, studies on solving algebraic word problems have not measured the faithfulness of a generated explanation. Existing studies so far measured the quality of explanations using plausibility only. Following Jacovi and Goldberg (2020), we define faithful explanation as one that accurately represents the reasoning process behind the model's solution equation. Humans expect an explanation to be faithful. However, a model can generate an explanation that may not be related to the equations (Jacovi and Goldberg, 2020); it can generate random plausible sentences independently from the process of generating solution equations. Therefore, measuring faithfulness is meaningful in that a highly faithful explanation reflects a solution equation generation process that is expected by human problem solvers.

3 The EPT-X Model

The proposed model, Expression-Pointer Transformer with Explanations (EPT-X)[1], is a variant of Expression-Pointer Transformer (EPT; Kim et al. 2020), which is state-of-the-art *correctness* model. Figure 1 depicts the two phases EPT-X model. (1) *Plausibility*: In phase 1, EPT-X receives a problem text as an input and generates explanations for each number/variable. The number of variables is also predicted in this phase. (2) *Faithfulness*: In phase 2, EPT-X receives both the original problem and the generated explanations as inputs and then builds an equation using EPT. To jointly train these two phases, we add up the loss functions for the number

[1] http://github.com/snucclab/ept-x

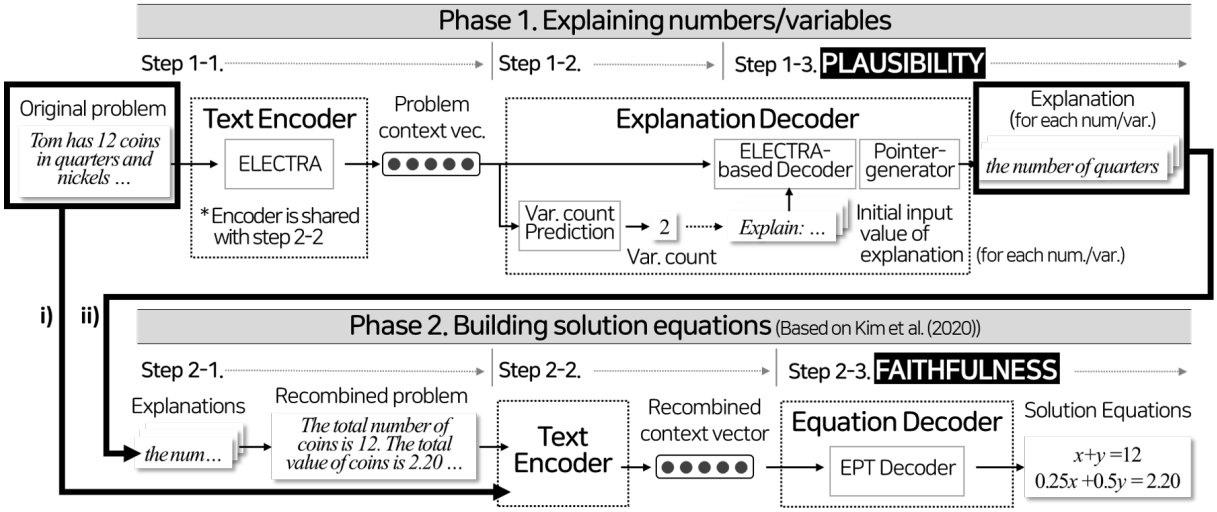

Figure 1: The two-phase pipeline of generating explanations and equations in our EPT-X model. i) shows the original problem input, and ii) shows the explanations which are recombined at Step 2-1.

of variables, explanations, and equations; all three use smoothed cross-entropy (Szegedy et al., 2016) with $\alpha = 0.01$.

3.1 Phase 1. Explaining numbers/variables

Phase 1 is a three-step procedure for generating explanations as shown in the top part of Figure 1. Phase 1 contains two components: text encoder and explanation decoder.

Step 1-1. Compute problem text vectors The text encoder receives a natural language problem as an input and computes problem context vectors. To utilize world knowledge in the computation process, we used ELECTRA (Clark et al., 2020), a pre-trained language model. After applying the text encoder, we obtain the problem context vector $\mathbf{w}_s$ for each token w_s in the given problem.

Step 1-2. Predict the number of variables Using the problem context vectors, EPT-X predicts the number of required variables N to solve the given problem. Using the first token's problem context vector $\mathbf{w}_0$, we compute the probability distribution of N as follows:

$$P(N) = \text{softmax}\left(\text{FF}_{n1}\left(\text{ReLU}\left(\text{FF}_{n2}(\mathbf{w}_0)\right)\right)\right),$$

where $\text{FF}(\cdot)$ indicates the feed-forward layer. We set the maximum number of variables to 9.

Step 1-3. Generating plausible explanations The explanation decoder then produces explanations using problem context vectors as memories. The decoder uses a Transformer (Vaswani et al., 2017) decoder and a pointer-generator network (See et al., 2017). Before predicting the next explanation token x_{t+1}, the Transformer decoder computes a hidden state $\mathbf{h}_t$ based on the problem context vectors $\mathbf{w}_s$ and previously generated explanation tokens $x_1, \cdots, x_t$. To utilize world knowledge in generating explanations, we adopt Rothe et al. (2020) and use ELECTRA (Clark et al., 2020) as the initial weight.

The pointer-generator head receives the computed $\mathbf{h}_t$ and predicts the next token. Let p_g, P_v, and P_c be the probability of using the generated word, the probability of generating from the vocabulary, and the probability of copying from the problem, respectively. Then, the next token x_{t+1} is predicted as follows:

$$\begin{aligned}
x_{t+1} &= \arg\max_{\omega} p_g P_v(\omega) + (1 - p_g) P_c(\omega), \\
p_g &= \sigma\left(\text{FF}_g\left(\mathbf{w}_t^* \oplus \mathbf{h}_t \oplus \text{E}(x_{t-1})\right)\right), \\
P_v(\omega) &= \text{softmax}\left(\text{FF}_v(\mathbf{h}_t)\right), \\
P_c(\omega) &= \textstyle\sum_{w_s : w_s = \omega} \text{attn}(\mathbf{w}_s, \mathbf{h}_t), \\
\mathbf{w}_t^* &= \textstyle\sum_{w_s} \text{attn}(\mathbf{w}_s, \mathbf{h}_t),
\end{aligned}$$

where $\sigma(\cdot)$, $\text{E}(\cdot)$, and $\text{attn}(\cdot)$ indicate the sigmoid, embedding, and single-head attention scoring function, respectively. And $\oplus$ indicates concatenation of vectors.

Plausibility of explanation is implemented during this stage by generating an explanation for each number/variable separately. We use unique initial input values for all numbers and variables. This method has been used in other studies to bind the decoder to a specific context (Raffel et al., 2020;

Keskar et al., 2019). For numbers, instead of using the initial input value '[CLS]' of the Transformer decoder, we use the input "[CLS] explain: *context* [SEP]," where the *context* part depends on the number or variable. For the numbers, we use a window of tokens that are near the given number token. For example, if the window size is three, we use three tokens placed before and after the given token. For variables, we use the variable index because variables do not appear in the problem. So, for example, the initial input value of the nth variable becomes "[CLS] explain: variable n [SEP]."

3.2 Phase 2. Building solution equations

Phase 2 is a three-step procedure for producing equations as shown in the bottom part of Figure 1. Phase 2 uses the same text encoder from Phase 1.

Step 2-1. Recombine explanations Inspired by human paraphrasing strategies (Conway and Polya, 1985; Gagnon and Maccini, 2001; Montague, 2008), EPT-X paraphrases the original problem by recombining its understanding. First, the model places each explanation and the corresponding number token value into a sentence: "*explanation* is a number *value*." for numbers and "What is *explanation*?" for variables. Then, EPT-X creates a recombined problem by concatenating these paraphrased sentences. We randomly recombined one of the reference explanations in the training process as EPT-X may not generate explanations ideally.

Step 2-2. Compute recombined context vectors The text encoder once again receives both the original problem and the recombined problem as inputs and computes the recombined context vectors $\mathbf{r}_i$ for each input token r_i. We designed EPT-X to use both problems for two reasons. First, using the original problem can avoid information loss. Second, using the recombined problem can make the equation decoder exploit the information of explanations. We arrange these two problems into the text encoder as follows: "[CLS] *original* [SEP] *recombined* [SEP]."

Step 2-3. Generate equations faithfully The equation decoder then produces equations using the recombined context vectors as memories. Following the EPT model (Kim et al., 2020), the decoder produces equations using expression tokens, each of which is a tuple of an operator and relevant operands. So, the equation decoder predicts the next jth expression as follows. First, the decoder receives expression tokens generated so far and converts them into embedding vectors $\mathbf{v}_k$ $(k = 0, \cdots, j-1)$. Then, using these embedding vectors $\mathbf{v}_k$ and recombined context vectors $\mathbf{r}_i$, the decoder builds an equation context vector $\mathbf{q}_j$ for the next expression. Lastly, the decoder simultaneously predicts the next operator and its required operands using $\mathbf{q}_j$. Thus, when we translate expressions into an equation, we can compute an answer to a problem.

The faithfulness of explanation is implemented during this stage by using explanations as the input data source. We change the input format of numbers and variables in EPT's equation decoder to use explanations. Originally, EPT used different types of vectors to input them: the encoder's hidden state for each known number and the decoder's hidden state for each unknown variable. However, in EPT-X, we guide the model to utilize the information from the explanation when writing an equation. As all numbers and variables appear in the recombined problem, EPT-X uses the vector $\mathbf{r}_i$ corresponding to each number/variable.

4 The PEN dataset

We release 'Problems with Explanations for Numbers' (PEN)[2], an algebraic word problem dataset with problem texts, equations, and explanations of numbers/variables for each problem to train and evaluate EPT-X. As existing datasets for algebraic word problems do not contain explanations, we provided explanations on the existing three benchmark datasets on solving algebraic word problems[3]: ALG514 (Kushman et al., 2014), DRAW-1K (Upadhyay and Chang, 2017), and MAWPS (Koncel-Kedziorski et al., 2016). The following sections introduce the two stages of building PEN: preparation for correcting errors and annotation for collecting explanations.

4.1 Preparation: correcting errors

We corrected the errors and organized the data in three steps. In the first step, we revised the problems' typos, grammatical errors, and logical flaws. For example, we found a problem asking

[2] http://github.com/snucclab/pen

[3] Though we considered using AQuA-RAT (Ling et al., 2017), which includes rationale about computation, we found that using it is intractable since we have to re-collect explanations for numbers and variables in most problems.

	Train	Dev.	Test	Total
Problems	2,581	365	365	3,581
Explanations	36,261	4,569	4,719	45,549
Words/Prob.	31.01	30.81	30.91	30.98
Num/Prob.	4.22	4.09	4.21	4.20
Var/Prob.	1.36	1.35	1.39	1.36
Words/Expl.	7.73	7.76	7.72	7.73

Table 2: Statistics of PEN dataset

about 'Senators' after telling a story about 'the House of Representatives.' So we replaced the out-of-context term with the other one. Second, we extracted numeric words from the modified text using WordNet (Fellbaum, 1998); Arabic numerals, fractions, ordinals, and their synonyms were extracted. Third, to normalize equations, we re-formulated them according to nine source formulas organized by Mayer (1981) and four formulas organized by Carpenter et al. (1996).

Among 3,886 problems from the three datasets, we corrected 3,581 problems in the PEN dataset. We excluded 305 problems because they are (1) exact duplicates of others (303 problems)[4] or (2) not an algebra problem (2 problems)[5]. After excluding 305 problems, we further revised incorrect equations: 62 of the 3,581 problems (1.73%).

4.2 Annotation: collecting explanations

When collecting natural language explanations, the explanations can be irrelevant to the given problem without any guidelines. Thus, we instructed our workers to follow eight rules, including "Use at least one word appearing in the problem text when writing an explanation." Moreover, we make workers obey the rules consistently using a web-based system. Details about all eight rules and the web-based system are illustrated in Appendix A.

Fourteen skilled workers provided explanations for numbers and variables in a problem. Before assigning workloads, we split the entire dataset into training (80%), development (10%), and test (10%) sets. Then, we collected multiple explanations for each problem; 3 for training set and 4 for the other. Table 2 shows the statistics of the PEN dataset. PEN has 45,549 explanations, and the average number of words in an explanation is 7.73.

[4]Since we manually corrected errors and flaws in each problem and combined three different datasets, some problems become exact duplicates of other problems.

[5]These problems cannot be solved with a multivariate equation alone: problems about least common multiples or counting the number of cases.

5 Experimental setup

To verify whether the EPT-X model can solve an algebraic problem correctly while generating plausible and faithful explanations, we conduct three types of analyses: model performance analysis, quantitative error analysis, and qualitative output analysis. This section illustrates each analysis and further implementation details.

5.1 Model performance analysis

The model performance analysis measures the model's correctness, which is the percentage of correctly answered problems on the PEN dataset. We regard an answer to be correct only if the answer values of all the variables in the problem are paired and solved correctly. For example, Table 1 shows that a correct answer contains two variables and answer values of $x = 8$ and $y = 4$. Existing studies regarded $x = 4$, $y = 8$ to be a correct solution(Kushman et al., 2014; Kim et al., 2020; Lee and Gweon, 2020). However, in the context of generating explanations along with solutions, different explanations are generated with different variables (Conway and Polya, 1985; Montague, 2008). Therefore, we enforce a stricter constraint that requires that a variable should be matched with a correct answer value.

Using the correctness, we compare the EPT-X model with two previous inexplainable models (EPT (Kim et al., 2020), GEO (Lee and Gweon, 2020)) and human performance. The EPT is a model that generates one expression at a time and uses pointers instead of classifiers, and it achieved state-of-the-art accuracy on MAWPS and DRAW datasets. The GEO is a model that mixes encoder and decoder outputs before predicting a token, and it achieved state-of-the-art accuracy on DRAW and ALG514 datasets. To establish a human performance baseline, our research team manually checked for the answer correctness of the original datasets of ALG514, DRAW, and MAWPS. We found that 62 of the 3581 problems were incorrectly solved, thus yielding a human baseline performance of 98%.

5.2 Quantitative error analysis

We conduct four types of error analyses to understand the possible cause of EPT-X solution errors:

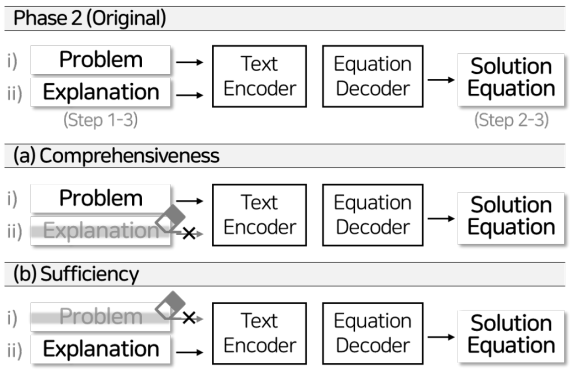

Figure 2: Two methods on measuring faithfulness, inspired by DeYoung et al. (2020)

plausibility test, faithfulness test, faithfulness control test, and error propagation test. Through these four types of analyses, we show how the generated explanations by the EPT-X model can be used to understand the equation generation process.

5.2.1 Plausibility test

To test the plausibility of explanations, we used BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), CIDEr (Vedantam et al., 2015), and BLEURT (Sellam et al., 2020). These metrics can measure the extent of similarity between a generated explanation and the reference explanation in the dataset.

Using the above four metrics, we compare the EPT-X model with a baseline model and human performance. First, we compare EPT-X with a baseline model which only contains Phase 1 (P1-only). This model can generate an explanation for each number/variable but cannot solve a word problem. Second, we compared EPT-X with human performance. While collecting explanations for each number/variable, we collected four sets of explanations. Of these four, one set is randomly set aside to serve as a hypothesis sentence and the other three as reference sentences when measuring human performance.

5.2.2 Faithfulness test

To measure the faithfulness of EPT-X, we used two metrics: sufficiency and comprehensiveness (DeYoung et al., 2020). First, in our context, *comprehensiveness* means "were explanations (Step 1-3) needed to produce the solution equation (Step 2-3)?". Figure 2 (a) shows the measurement setup for comprehensiveness. Specifically, we examined the amount of change between the two output solution equations: the equation from the original Phase 2 setup and the equation that is generated with only the problem text for the input of Phase 2. Since the output equation is not a single prediction as in DeYoung et al. (2020), we measured the change in the solution equations using tree edit distance (Zhang and Shasha, 1989).

Secondly, in our context, *sufficiency* means "do explanations (Step 1-3) contain enough information to produce solution equation (Step 2-3)?". Figure 2 (b) shows the measurement setup for sufficiency. Here, we examined the amount of change between the two output solution equations: the equation from the original Phase 2 setup and the equation that is generated with only the generated explanation. Similar to the comprehensiveness measure, the difference in equations was also computed using tree edit distance.

To provide a statistical baseline for interpreting the two metrics of comprehensiveness and sufficiency, we adopted a bootstrapping method (Koehn, 2004). We sampled 500 bootstrapped samples (each sample has 50 problems) to estimate the population distribution of each metric. After the estimation, we conducted hypothesis testing for each metric. For comprehensiveness $\mathcal{C}$, we set the following hypothesis $H_A : \mathcal{C} > 1$ as we expect to observe changes in equations when using only the problem input compared to using both problem and explanation input. For sufficiency $\mathcal{S}$, we set the following hypothesis $H_A : \mathcal{S} < 1$ as we expect to observe no change in equations when only using the explanation input.

5.2.3 Faithfulness control test

To examine a trade-off relationship between correctness and faithfulness, we train and analyze two variants of EPT-X, whose faithfulness is controlled. The first model is an inherently *faithful* model (EPT-XF) that uses only the explanation, but not the original problem, as an input to Phase 2. As EPT-XF entirely depends on the explanation to generate a solution equation, the model passes the test of faithfulness by definition. The second model is an inherently *unfaithful* model (EPT-XU) that uses only the original problem, but not the explanation from phase 1, as an input to Phase 2. As EPT-XU ignores the explanation input, the model fails the test of faithfulness by definition. Implementation details on these two models are explained in Appendix B.

5.2.4 Error propagation test

To examine how the quality of explanation affects the model's correctness, we used two models, EPT-X and EPT-XF. Both models employ a two-phase architecture, thus they are prone to errors in both phases. For the error propagation test, we examine how the performance of Phase 1 (plausibility) affects the end-task performance (correctness). Note that testing EPT-X may not reveal the errors that are solely propagated from the generated explanation because EPT-X also uses the original problem as an input. Therefore, EPT-XF performance was also measured in order to examine the impact of errors from the generated explanation only.

We measured the amount of error propagation in the two models, EPT-X and EPT-XF, by comparing correctness under two conditions: control and experiment. Under the control condition, the models build solution equations based on explanations generated by themselves. On the other hand, under the experiment condition, they build solution equations based on the gold standard explanations. Then, we measure the change of correctness between these two conditions for each model. Here, we expect that the change to reveal the proportion of problems affected by errors that are propagated from Phase 1.

5.3 Qualitative output analysis

The explanations generated by EPT-X were analyzed qualitatively using two methods. First, to measure the quality of the generated explanation itself, we manually labeled the quality in the PEN's development set using two criteria: (1) *plausibility* and (2) *faithfulness*. Human coders were asked to label an explanation to be plausible when the explanation and the original problem text are coherent in meaning. And for faithfulness, we asked human coders to build a solution equation using only the explanation produced from the EPT-X model. If the generated solution equation is identical to the EPT-X generated solution equation, the explanation is labeled to be faithful.

Second, to find the primary cause of errors when generating an explanation, we manually classified errors in EPT-X's explanations. The errors were categorized by comparing the generated explanation with the gold-standard explanation. We also used the PEN development set for this analysis.

		Dev.	Test
Human		98.35	98.35
Baselines:	EPT	77.26	74.52
	GEO	63.01	62.47
Proposed:	EPT-X	72.88	69.59

Table 3: Correctness of EPT-X on PEN dataset

5.4 Implementation Details

We describe three major implementation details used for training EPT-X: encoder, optimizer, and training epochs. For the text encoder component, EPT-X uses the base version of ELECTRA (Clark et al., 2020). We fixed its embedding and tied the embedding with the weights of FF_v in the explanation decoder to preserve the world knowledge in the embedding and to stabilize the training procedure. For the optimizer, we used LAMB (You et al., 2020) with a learning rate of 0.00176, which was found from a grid search on the development set. Finally, for the training epochs, we trained EPT-X for 500 epochs. Appendix C lists additional details of the model, including hardware, software, libraries, hyper-parameters, and random seeds.

6 Result and Discussion

The result of three analyses reveals that the EPT-X can generate an equation correctly based on a plausible and faithful explanation. First, Section 6.1 presents the result of the model performance analysis, which shows that EPT-X can achieve correctness 5% lower than previous inexplainable models. Second, Section 6.2 shows the result of error analysis, which reveals that many of EPT-X's errors are due to insufficient explanations. And lastly, Section 6.3 shows the result of qualitative analysis, which reveals three types of errors found in the explanation generation process of EPT-X.

6.1 Model performance analysis

The model performance analysis shows that EPT-X generates equations with 69.59% accuracy on the PEN dataset, despite being a two-phase model. Table 3 shows that adding the explanation generation functionality decreases the accuracy by approximately 5%, compared to state-of-the-art model EPT [6]. We suspect that this performance

[6]The results on the whole dataset is reported in this section, whereas results on each subset are reported in Appendix D.

		BLEU	ROUGE	CIDEr	BLEURT
Dev:	Human	57.16	78.66	343.0	71.44
	P1-Only	60.26	78.02	346.7	69.11
	EPT-X	60.07	77.99	347.1	69.61
Test:	Human	55.69	78.28	347.3	71.51
	P1-Only	59.32	77.99	342.9	69.69
	EPT-X	60.49	78.34	341.5	69.59

Table 4: Plausibility of EPT-X on PEN dataset

	Dev.	Test	H_A
Comprehensiveness	5.97^{**}	6.56^{**}	($\mathcal{C} > 1$)
Sufficiency	1.20	1.19	($\mathcal{S} < 1$)

$^{*}\ p < 0.05$, $^{**}\ p < 0.01$

Table 5: Mean faithfulness of EPT-X on PEN dataset

drop is due to propagation of the errors in the generated explanations. Regardless, the results of EPT-X are meaningful since the model automatically generates explanations of problems without too much decrease in correctness. The difference of 5% is quite promising compared to that of the previous explainable model (Ling et al., 2017), about 40%, although a direct comparison is not possible due to differences in datasets.

6.2 Quantitative error analysis

6.2.1 Plausibility test

EPT-X achieved plausibility scores that are comparable to humans and the P1-only model, as shown in Table 4. The differences in plausibility scores between EPT-X and the other two baselines range between 1 to 2 points. This result indicates that EPT-X can select proper words to generate an explanation. In fact, BLEU-4 score of 60 is promising compared to Ling et al. (2017) (27.2). Given that EPT-X achieved human-level performance in terms of plausibility, but not for correctness, we explored the faithfulness metric to examine additional causes for the low model performance.

6.2.2 Faithfulness test

The results of the faithfulness test showed two characteristics of the explanation output of EPT-X. In terms of comprehensiveness, the generated explanation contains some information required to generate a solution equation, as evidenced by Table 2. Here, we observe that EPT-X passed the comprehensiveness test for the 99% confidence level. That is, compared to using both inputs in Phase 2, forcing EPT-X to use only the original problem input made EPT-X generate a different solution equation. This result suggests that the generated explanation provides information, which is not provided by the original problem but contributes to generating a solution equation.

Meanwhile, in terms of sufficiency, the generated explanation may not provide sufficient numeric information to generate a solution equation. Table 2 shows that EPT-X failed the sufficiency test under the confidence level of 95%. That is, EPT-X generates different solution equations when it only receives the generated explanation as input in Phase 2. So, the explanation generated in Phase 1 does not contain sufficient information, which is contained in the original problem, to generate a solution equation.

		Dev.	Test
Proposed:	EPT-X	72.88	69.59
Variants:	EPT-XF	66.03	62.19
	EPT-XU	76.16	73.70

Table 6: Result of faithfulness control test

6.2.3 Faithfulness control test

As the generated explanation fails to capture some information from the original problem, the correctness may change when we control the faithfulness of a model. Specifically, the control test shows that there is a trade-off between faithfulness and correctness; as faithfulness increases, the correctness decreases. Table 6 shows that the most faithful model EPT-XF achieves the lowest correctness score, which is 6% lower than EPT-X. Conversely, the most unfaithful model EPT-XU achieves the highest correctness score, which is 4% greater than EPT-X. Thus, the results of the faithfulness test and the faithfulness control test imply that in order to achieve a higher correctness score, we should verify whether the explanation contains “sufficient" information to build a correct equation.

6.2.4 Error propagation analysis

The error propagation test shows that the generated explanation does not contain sufficient information to build a correct equation for some problems. Table 7 shows that both EPT-X and EPT-XF can outperform the EPT model by 8% when using a gold standard explanation as an input. However,

		Generated	Gold	Change
Dev.:	EPT-XF	66.03	83.29	+17.26
	EPT-X	72.88	86.03	+13.15
Test:	EPT-XF	62.19	85.20	+23.01
	EPT-X	69.59	85.21	+15.62

Table 7: Result of error propagation test of explanation

using explanations generated by the EPT-X model may decrease the correctness by more than 15%. That is, more than 15% of errors are due to information loss in Phase 1.

6.3 Qualitative output analysis

Quality of explanations: The qualitative analysis showed that the quality of the generated explanations could be improved given that information required for solving word problems is missing. When we manually labeled the generated explanations for plausibility, 167 of 365 problems (45.8%) were labeled as plausible. Thus, the majority of the explanations are insufficient or contain incorrect information to generate a correct equation. Similarly, when we manually labeled the generated explanations for faithfulness, 201 of 365 problems (55.1%) were labeled as faithful. That is, when the same yet insufficient explanations were used to generate equations, the equations generated by EPT-X and humans were different.

Three categories of errors: Additional qualitative analysis found three possible causes for the EPT-X errors. Here, we will briefly discuss the causes, and the detailed examples are illustrated in Appendix E. First, when a problem mentions several entities with similar properties (e.g., Heather's weight and Emily's weight), the difference between entities is ignored in the encoder (69 of 118 incorrect problems; 58.5%). This error implies that the context window used in Step 1-3 may not be big enough to distinguish two different entities. Second, if a problem provides multiple situations related to an entity (e.g., outward trip versus return trip), assigning a corresponding number to the correct situation fails in the encoding process. Detailed explanations of situations of a word problem were often omitted in the encoder (57 of 118 problems; 48.3%). Third, when a problem contains some irrelevant numbers, which are not used in solving the problem (e.g., year), sometimes an explanation for an irrelevant number was generated instead of the relevant one in the encoding process (32 of 118 problems; 27.1%). The second and third error types imply that sharing the encoder in Phases 1 and 2 might have caused confusion. The goal of the encoder in Phase 1 was to provide a detailed explanation of a given number, whereas the goal in Phase 2 was to build an equation, which involves ignoring some details to build an abstraction in the form of an equation.

7 Conclusion

This study proposed a novel neural model EPT-X, Expression Pointer Transformer with Explanations, which generates explanations along with solution equations. The EPT-X model was designed to address two criteria of *plausibility* and *faithfulness* when generating an explanation. To address plausibility, the model generates explanations for each number/variable in the solution equation separately. And to address faithfulness, the model produces equations based on the information in the generated explanation. In addition to EPT-X, we release a new dataset, Problem with Explanations for Numbers (PEN), which extends existing three algebraic word problem datasets by augmenting explanations for numbers/variables. Using the PEN dataset, we conducted three analyses. The model performance analysis revealed that EPT-X could produce a correct equation with 69.59% accuracy. The quantitative error analysis showed that the EPT-X model could produce a plausible albeit insufficient explanation. Lastly, the qualitative output analysis identified three categories of errors made when generating explanations. Despite the insufficiency of explanations generated by the EPT-X model, our work is significant in that we demonstrated the possibility of generating explanations while solving an algebraic word problem. For future work, we plan to improve the correctness and faithfulness of EPT-X to enhance the existing state-of-the-art model.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant (No. 2020R1C1C1010162) and the Institute for Information & communications Technology Promotion (IITP) grant (No. 2021-0-02146), both funded by the Korean government (MSIT). Also, we thank S. Oh, Y. Lee, J. An, J. Kim, and H. Rhim who helped in building the PEN dataset.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas P Carpenter, Elizabeth Fennema, and Megan L Franke. 1996. Cognitively guided instruction: A knowledge base for reform in primary mathematics instruction. *The elementary school journal.*, 97(1):3–20.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2656–2668, Minneapolis, Minnesota. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

John H Conway and G. Polya. 1985. *How to solve it*, volume 85. Princeton university press.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Christine Fellbaum. 1998. *WordNet: an electronic lexical database*. MIT Press.

Joseph Calvin Gagnon and Paula Maccini. 2001. Preparing students with disabilities for algebra. *TEACHING Exceptional Children*, 34(1):8–15.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Asha K. Jitendra, Edward Sczesniak, Cynthia C. Griffin, and Andria Deatline-Buchman. 2007. Mathematical word problem solving in third-grade classrooms. *The Journal of Educational Research*, 100(5):283–302.

Asha K. Jitendra and Jon R. Star. 2012. An exploratory study contrasting high- and low-achieving students' percent word problem solving. *Learning and Individual Differences*, 22(1):151–158.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Kyung Seo Ki, Donggeon Lee, Bugeun Kim, and Gahgene Gweon. 2020. Generating equation by utilizing operators : GEO model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 426–436, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Point to the Expression: Solving Algebraic Word Problems using the Expression-Pointer Transformer Model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3768–3779, Online. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

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.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

D. Lee and G. Gweon. 2020. Solving arithmetic word problems with a templatebased multi-task deep neural network (t-mtdnn). In *2020 IEEE International*

Conference on Big Data and Smart Computing (BigComp), pages 271–274.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Richard E. Mayer. 1981. Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *Instructional Science*, 10(2):135–175.

Marjorie Montague. 2008. Self-regulation strategies to improve mathematical problem solving for students with learning disabilities. *Learning Disability Quarterly*, 31(1):37–44.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3082–3088. AAAI Press.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504, Valencia, Spain. Association for Computational Linguistics.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306, Austin, Texas. Association for Computational Linguistics.

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 *Advances in neural information processing systems*, pages 5998–6008.

R. Vedantam, C. L. Zitnick, and D. Parikh. 2015. Cider: Consensus-based image description evaluation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4566–4575.

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.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, Lisbon, Portugal. Association for Computational Linguistics.

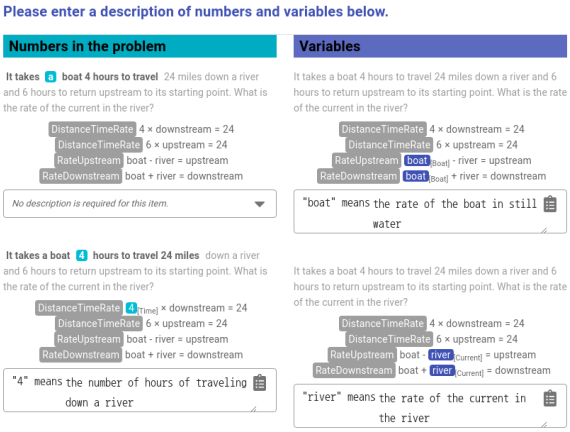

Figure 3: A screenshot of the system used for annotating explanations on the PEN dataset

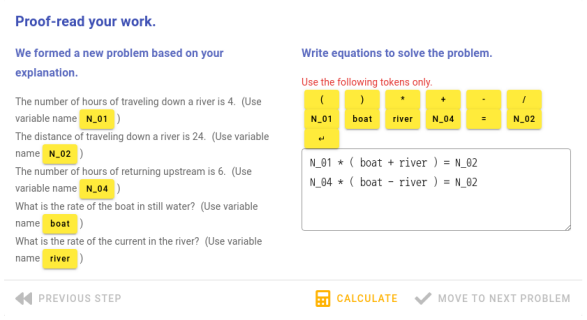

Figure 4: A screenshot of the system used for validating explanations on the PEN dataset

A Annotating explanations for PEN dataset

This section describes the detailed process of annotating explanations. Using a web-based system shown in Figure 3, coders inputted a natural language explanation for each number/variable. To provide situational information for each number/variable, we highlighted text snippets and equations related to the target number/variable. Based on the given information, the coder needed to complete the following sentence: "*number* means ..."

As the coders input natural language explanations, a coder's explanation may not be coherent with the given word problem. To make the explanation coherent with the problem, we used two strategies: rules and validation. For the rules, we instructed the coders to follow the eight rules below:

Rule 1. Please write an explanation of the situation that the number/variable denotes, using the words appearing in the text.

Rule 2. Each explanation is a simple noun phrase that has 3 to 25 words. Try to be concise.

Rule 3. Use at least one word appearing in the problem text when writing an explanation.

Rule 4. Do not use the same explanation for different objects.

Rule 5. You should be able to formulate equations for solving the problem, using your explanations only.

Rule 6. We suggest writing a difference A-B as "the value of A minus B."

Rule 7. We suggest writing a ratio A/B as "the ratio of A to B."

Rule 8. We suggest writing a numerator[denominator] of A/B as "the numerator[denominator] of the ratio of A to B."

Moreover, to assist the coders in obeying the eight rules, the system consistently checked whether the coders followed the rules. If one of the first four rules is broken, the system mandates the coder to obey the broken rule before proceeding to the next problem. For the other four rules, the system shows hints to make the coder manually verify the rules.

For the validation, we asked coders to validate their work by solving a problem reconstructed from the annotated explanations. For example, Figure 4 shows that the system synthesizes a problem by concatenating a coder's explanations and requests the coder to solve the synthesized problem. The coder can proceed to the next problem if an answer to the synthesized problem is the same as the original problem.

B Two variants of EPT-X

To investigate how the faithfulness of a model affects its correctness, we designed two variants of EPT-X: EPT-XF and EPT-XU. The following paragraphs illustrate each model.

EPT-XF is an inherently faithful variant of EPT-X. This model utilizes only the recombined problem in Step 2-2. So, the text encoder receives the following input in Step 2-2: "`[CLS]` *recombined problem* `[SEP]`." As the recombined problem contains information required in Step 2-3, the

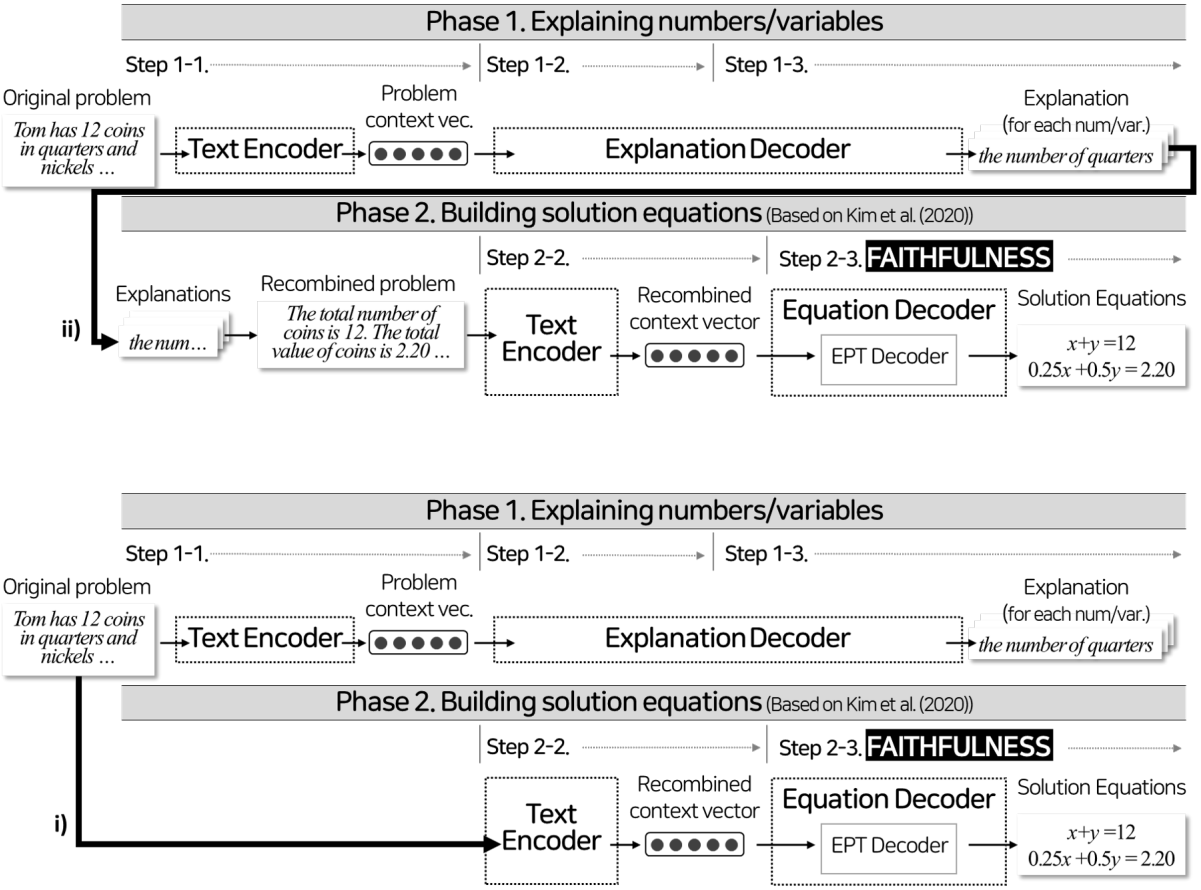

Figure 5: Two variants of EPT-X model: EPT-XF (top) and EPT-XU (bottom).

Model	Datasets				# of
	PEN	ALG	DRAW	MAWPS	Param.
EPT	.00088	.00176	.00176	.00125	122M
EPT-X	.00176	.00176	.00176	.00088	263M
EPT-XF	.00176	.00088	.00088	.00176	263M
EPT-XU	.00176	.00125	.00176	.00125	263M

Table 8: Selected learning rates and training information for the EPT and EPT-X model. Best rates are selected using the development split on PEN and DRAW dataset, and fold 0 split for the others.

other steps are unchanged. Note that, in this model, the model is more prone to errors in the generated explanation since the equation decoder solely depends on the output of the explanation decoder.

EPT-XU is an inherently unfaithful variant of EPT-X. This model utilizes only the original problem in Step 2-2. So, the text encoder receives the following input in Step 2-2: "`[CLS]` *original problem* `[SEP]` X_0 X_1 $\cdots$ X_N `[SEP]`" where N is the predicted number of variables from Step 1-2. We concatenated 'X_0 X_1 $\cdots$ X_N' in order to keep Step 2-3 unchanged. Step 2-3 requires vectors representing either a written number or a required variable to predict an operand. However, if we remove the recombined problem from the input of Step 2-2, the output of Step 2-2 could not provide such vectors, especially for the variables. Thus, for each variable, the list of variables is added to produce a vector value that Step 2-3 can use. And, for each number, the vector corresponding to the number written in the original text is used in Step 2-3 as the EPT model did.

C Implementation details

In this section, we describe the implementation details of EPT-X.

- Hardware:

  CPU: AMD Ryzen Threadripper 3970X
  GPU: GeForce RTX 3090, four cards
  Memory: 192GB

- Software:

  OS: Ubuntu 20.04.2 LTS (kernel 5.4.0-80)
  CUDA: 11.1
  Graphic Driver: 460.73.01
  Python: 3.8.10 (with virtualenv)

- Python libraries:

  PyTorch 1.8.1+cu111
  transformers 4.6.1 (for ELECTRA)
  torch-optimizer 0.1.0 (for LAMB)
  ray 1.3.0 (for hyperparameter search with ray[tune])
  bleurt `git+https://github.com/google-research/bleurt` Commit `c6f2375` (Oct 15th, 2021; for BLEURT)
  tensorflow 2.7.0 (for BLEURT)
  numpy 1.21.0
  scipy 1.7.0
  sympy 1.8
  pycocoevalcap 1.2
  pycocotools 2.0.2
  zss 1.2.0

- Hyperparameters and options for EPT-X:

  text encoder: google/electra-base-discriminator. We fixed the embedding layer to preserve the world knowledge in the embedding and to stabilize the training procedure.
  explanation decoder: Following (Rothe et al., 2020), we inserted randomly initialized cross-attention layers in an ELECTRA model of google/electra-base-discriminator.
  equation decoder: 6 layers of Transformer decoder. We tied weights across these layers.
  training epoch: 500
  optimizer: LAMB (You et al., 2020) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-12}$.
  learning rate: To find the best learing rate for each dataset and model, we conducted grid-search. Among the possible learning rates in $\{0.00088, 0.00125, 0.00176, 0.0025\}$, we selected a model with the highest answer correctness. For P1-Only models, we used EPT-X's learning rate. Table 8 shows the selected learning rates. Also, we applied linear warm-up for 10 epochs and linear decay for the rest of the epochs.

	ALG514	DRAW	MAWPS
Total counts			
Problems	514	998	2,372
Explanations	8,493	15,291	24,852
Average across problems			
Words	38.82	32.70	27.92
Numbers	5.78	4.96	3.41
Variables	1.82	1.85	1.02
Words/Expl.	7.61	8.32	7.37

Table 9: Statistics of PEN's subsets

	MAWPS	DRAW	ALG514
Human	98.78	96.70	100.0
EPT	88.70	63.5	73.91
GEO*	84.51	62.5	82.1
EPT-X	84.57	56.0	67.07

* Copied from the published result.

Table 10: Correctness of EPT-X on PEN's subsets

window size in Step 1-3: 3 tokens
batch size: 16 problems per batch
BLEURT checkpoint: BLEURT-20-D6

To ensure the reproducibility of our experiment, we used separate random number generators with seed '1' in the following places:

- Code where building mini-batches for training
- Code where selecting gold set explanations randomly for training Phase 2
- Code where inputting gold set explanations for the error propagation analysis

D Result of correctness, plausibility, and faithfulness on subsets

Correctness: The PEN dataset contains three subsets corresponding to each benchmark dataset: ALG514, DRAW, and MAWPS. To make the performance on these datasets be compatible with previous benchmarks, we retained duplicated problems in these subsets. Table 9 also shows the statistics of these three subsets. DRAW is the most difficult subset to generate explanations since its explanation is the longest (8.32 words) among the three subsets while its text is the shortest (32.70 words). Likewise, MAWPS is the easiest subset among the three subsets.

	BLEU	ROUGE	CIDEr	BLEURT
MAWPS subset				
Human	54.80	79.07	334.7	70.81
EPT-X	79.36	88.01	448.0	82.12
DRAW subset				
Human	58.04	79.81	368.5	73.92
EPT-X	58.01	75.32	314.5	66.77
ALG514 subset				
Human	56.57	77.82	346.9	76.76
EPT-X	56.25	75.49	310.0	67.48

Table 11: Plausibility of EPT-X on PEN's subsets

	MAWPS (fold 0)	DRAW (test)	ALG514 (fold 0)
Sufficiency	0.69^{+}	1.74	1.55
Comprehensive.	4.83^{**}	9.57^{**}	10.33^{**}

$^{+}\ p < 0.1$, $^{*}\ p < 0.05$, $^{**}\ p < 0.01$

Table 12: Mean faithfulness of EPT-X on PEN's subsets

For each subset, we conducted the same comparative analysis to evaluate EPT-X. Tables 10 to 12 shows the EPT-X's performance on these subsets. Note that as we manually corrected problems and equations in PEN, the results cannot be directly compared with previous state-of-the-art models.

Table 10 reveals that EPT-X's correctness is comparable to the inexplainable models when generating explanations is simple. On the simplest subset MAWPS, EPT-X achieved answer correctness of 84.57%, which is 4% lower than to EPT (88.7%). Similarly, on the most difficult subset DRAW, EPT-X achieved an answer correctness of 56.0%, which is 7.5% lower than EPT (63.5%). As we discussed in Section 6, this performance decrease may be due to error propagation. On a subset whose explanation is difficult to generate (such as DRAW or ALG514), the chance of generating incorrect explanation increases. So, EPT-X can be swayed by wrong explanations considering the model's dependency on explanation.

Plausibility: Table 11 illustrates that EPT-X's plausibility scores are comparable to humans. On

Case 1. Encoder is confused an entity with others (69 of 118 problems)

Q. The Sears tower in Chicago is 1450 feet tall. The John Hancock center in Chicago is 1127 feet tall. Suppose you are asked to build a small-scale replica of each. If you make the Sears tower 3 meter tall, what would be the approximate height of the John Hancock replica?

Gold-standard	**EPT-X**
"How tall **Sears tower** is" is 1127.	"The height of the **Sears tower** in meter" is 1127.
"How tall **Hancock center** is" is 1450.	"The height of the **Sears tower** in meter" is 1450.
"Height of the **Sears tower replica**" is 3.	"The height of the **Sears tower**" is 3.
What[x_0] is "the height of Hancock replica?"	What[x_0] is "the height of the Sears tower?"
	What[x_1] is "the height of John Hancock center?"
Equation: $1127/1450 = 3/x_0$	**Equation:** $x_1 = 3 \times (1127 + 1450)$

Case 2. Encoder forgets to explain detailed situations (57 of 118 problems)

Q. Juan drives to work. Because of traffic conditions, he averages 22 miles per hour. He returns home, averaging 32 miles per hour. The total travel time is 2.25 hours. Write and solve an equation to find the time Juan spends driving to work.

Gold-standard	**EPT-X**
"The speed of Juan **driving to work**" is 22.	"The speed of Juan" is 22.
"The speed **returning home**" is 32.	"The speed of Juan"" is 32.
"The total travel time" is 2.25.	"The total travel time"" is 2.25.
What[x_0] is "the time traveled **to work**?"	What[x_0] is "the time Juan rowe spends?"
What[x_1] is "the time **returning from work**?"	
Equation: $22x_0 = 32x_1$	**Equation:** $22x_0 = 32x_0$
$2.25 = x_0 + x_1$	

Case 3. Encoder fails to identify numbers required to solve a problem (32 of 118 problems)

Q. There are 48 erasers in the drawer and 30 erasers on the desk. Alyssa placed 39 erasers and 45 rulers on the desk. How many erasers are now there in total?

Gold-standard	**EPT-X**
"The number of erasers in the drawer" is 48.	"The number of erasers in the drawer" is 48.
"**The number of erasers on the desk**" is 30.	
"The number of erasers added on the desk" is 39.	"The number of erasers placed on the desk" is 39.
What[x_0] is "the total number of erasers?"	What[x_0] is "the total number of erasers?"
Equation: $x_0 = 48 + 30 + 39$	**Equation:** $x_0 = 48 + 39$

Table 13: Three representative erroneous cases of EPT-X

the ALG514 and DRAW subsets, EPT-X showed slightly lower but comparable scores: at most 1% lower on BLEU, 2-5% lower on ROUGE, about 30-50 less on CIDEr, and 7-9% less on BLEURT. Meanwhile, on the MAWPS subset, EPT-X showed much higher plausibility scores than humans: about 20% more on BLEU, 8-10% more on ROUGE, about 110 more on CIDEr, and 11% more on BLEURT. These results imply that EPT-X could quickly learn how to generate explanations as the MAWPS dataset has more examples with simpler explanations than the other two subsets.

Faithfulness: Table 12 implies that the generated explanation is insufficient to produce a correct equation. EPT-X model passed both tests only on MAWPS subset, which has simpler explanations than the other two subsets. On the other hand, the model only passed the comprehensiveness test on DRAW and ALG514 subsets. This result implies that though EPT-X can generate a simple explanation appropriately, EPT-X may forget to explain some essential information when the target explanation is complicated.

E Example error cases

Table 13 shows the three error cases of EPT-X. First, Case 1 shows that the encoder is often confused with an entity to others. In this example, the model mistakenly equated 'Sears tower' and 'John Hancock center.' So, the EPT-X cannot utilize the concept of reduced scale, which is a key concept to solve the given problem. Second, Case 2 shows that the encoder often forgets to explain detailed situations of a word problem. In this example, the model unified two different situations: (1) a situation that Juan drives to work and (2) a situation that Juan returns home. So, the EPT-X cannot utilize the concept of round trip, which is a key concept to write the second equation, $2.25 = x_0 + x_1$. Lastly, Case 3 shows that the encoder sometimes fails to identify whether a number is significant to solve a word problem or not. In this example, the model did not describe the second number, 30. Without explaining the number, it is not possible to count the number of erasers correctly.