Adversarial Examples for Evaluating Math Word Problem Solvers

Vivek Kumar[*], **Rishabh Maheshwary**[*] **and Vikram Pudi**
Data Sciences and Analytics Center, Kohli Center on Intelligent Systems
International Institute of Information Technology, Hyderabad, India
{vivek.k, rishabh.maheshwary}@research.iiit.ac.in, vikram@iiit.ac.in

Abstract

Standard accuracy metrics have shown that Math Word Problem (MWP) solvers have achieved high performance on benchmark datasets. However, the extent to which existing MWP solvers truly understand language and its relation with numbers is still unclear. In this paper, we generate adversarial attacks to evaluate the robustness of state-of-the-art MWP solvers. We propose two methods *Question Reordering* and *Sentence Paraphrasing* to generate adversarial attacks. We conduct experiments across three neural MWP solvers over two benchmark datasets. On average, our attack method is able to reduce the accuracy of MWP solvers by over 40 percentage points on these datasets. Our results demonstrate that existing MWP solvers are sensitive to linguistic variations in the problem text. We verify the validity and quality of generated adversarial examples through human evaluation.

Original Problem
Text: Tim has 5 books. Mike has 7 books. How many books do they have together?
Equation: X = 5+7
Question Reordering
Text: How many books do they have together given that Tim has 5 books and Mike has 7 books.
Equation: X = 5*7
Sentence Paraphrasing
Text: Tim has got 5 books. There are 7 books in Mike's possession. How many books do they have?
Equation: X = 5*5

Table 1: A MWP and generated adversarial examples by our methods. Red and blue color denote the subject and the entity respectively of numerical values.

1 Introduction

A Math Word Problem (MWP) consists of a natural language text which describes a world state involving some known and unknown quantities. The task is to parse the text and generate equations that can help find the value of unknown quantities. Solving MWP's is challenging because apart from understanding the text, the model needs to identify the variables involved, understand the sequence of events, and associate the numerical quantities with their entities to generate mathematical equations. An example of a simple MWP is shown in Table 1. In recent years, solving MWPs has become a problem of central attraction in the NLP community. There are a wide variety of MWPs ranging from simple linear equations in one variable (Koncel-Kedziorski et al., 2016; Miao et al., 2020) to complex problems that require solving a system of equations (Huang et al., 2016; Saxton et al., 2019). In this paper, we consider simple MWPs which can be solved by a linear equation in one variable.

Existing MWP solvers can be categorized into statistical learning based (Hosseini et al., 2014; Kushman et al., 2014) and deep learning based solvers . However, recent deep learning based approaches (Wang et al., 2017; Xie and Sun, 2019; Zhang et al., 2020b) have established their superiority over statistical learning based solvers. Here, we will briefly review some recent MWP solvers. Initially, (Wang et al., 2017) modelled the task of MWP as a sequence to sequence task and utilized Recurrent Neural Nets (RNNs) to learn problem representations. Building upon this, (Chiang and Chen, 2018) focused on learning representations for mathematical operators and numbers, (Xie and Sun, 2019; Wang et al., 2019) utilized tree structure to develop decoders for MWP solvers. More recently, to learn accurate relationship between numerical quantities and their attributes (Zhang et al., 2020b) modelled encoder as a graph structure.

All such MWP solvers have achieved high per-

[*]Equal Contribution

formance on benchmark datasets. However, the extent to which these solvers truly understand language and numbers remains unclear. Prior works on various NLP tasks have shown that Deep Neural Networks (DNNs) attend to superficial cues to achieve high performance on benchmark datasets. Recently, (Patel et al., 2021) proposed a challenge test set called SVAMP which demonstrate that existing MWP solvers rely on shallow heuristics to achieve high performance. Instead of relying on standard accuracy metrics, many works have used adversarial examples (Szegedy et al., 2013; Papernot et al., 2017) to evaluate the robustness of neural NLP models. Adversarial examples are generated by making small changes to the original input such that the adversarial example is (1) semantically similar to the original input, (2) is grammatically correct and fluent and (3) deceives the DNNs to generate an incorrect prediction.

In (Jia and Liang, 2017) authors crafted adversarial attacks to test the robustness of QA systems. Prior works in (Glockner et al., 2018; McCoy et al., 2019) uses adversarial examples to show deficiencies of NLI models. Similarly, (Dinan et al., 2019; Cheng et al., 2019) uses adversarial examples to develop robust dialogue and neural machine translation models. Recently, there has been a plethora of work (Ebrahimi et al., 2017; Alzantot et al., 2018; Jin et al., 2020; Maheshwary et al., 2021, 2020) to evaluate text classification systems against adversarial examples. Although adversarial examples are commonly used for various NLP tasks, there has been no work that uses adversarial examples to evaluate MWP solvers. In this paper, we bridge this gap and evaluate the robustness of state-of-the-art MWP solvers against adversarial examples.

Generating adversarial attacks for MWP is a challenging task as apart from preserving textual semantics, numerical value also needs to be preserved. The text should make mathematical sense, and the sequence of events must be maintained such that humans generate the same equations from the problem text. Standard adversarial generation techniques like synonym replacement (Alzantot et al., 2018) are not suitable for MWP as the fluency of the problem statement is not preserved. Similarly, paraphrasing techniques like back-translation (Mallinson et al., 2017) are not ideal as they generate syntactically uncontrolled examples.

We propose two methods to generate adversarial examples on MWP solvers, (1) Question Reordering — It transforms the problem text by moving the question part to the beginning of the problem and (2) Sentence Paraphrasing — It paraphrases each sentence in the problem such that the semantic meaning and the numeric information remains unchanged. Our results demonstrate that current solvers are not robust against adversarial examples as they are sensitive to minor variations in the input. We hope that our insights will inspire future work to develop more robust MWP solvers. Our contributions are as follows:

1. To the best of our knowledge, this is the first work that evaluates the robustness of MWP solvers against adversarial attacks. We propose two methods to generate adversarial examples on three MWP solvers across two benchmark datasets.

2. On average, the generated adversarial examples are able to reduce the accuracy of MWP solvers by over 40%. Further, we experiment with different type of input embeddings and perform adversarial training using our proposed methods. We also conducted human evaluation to ensure that the generated adversarial examples[1] are valid, semantically similar and grammatically correct.

2 Proposed Approach

2.1 Problem Definition

A MWP is defined as an input of n tokens, $\mathcal{P} = \{w_1, w_2..w_n\}$ where each token w_i is either a numeric value or a word from a natural language. The goal is to generate a valid mathematical equation $\mathcal{E}$ from $\mathcal{P}$ such that the equation consists of numbers from $\mathcal{P}$, desired numerical constants and mathematical operators from the set $\{/, *, +, -\}$. The above problem can also be expressed as $\mathcal{P} = \{S_1, S_2..S_k, Q\}$ where Q is the question, $\{S_1, S_2..S_k\}$ are the sentences constituting the problem description.
Let $\mathbf{F} : \mathcal{P} \rightarrow \mathcal{E}$ be a MWP solver where $\mathcal{E}$ is the solution equation to problem $\mathcal{P}$. Our goal is to craft an adversarial text input $\mathcal{P}^*$ from the original input $\mathcal{P}$ such that the generated sequence is (1) semantically similar to the original input, (2) preserves sequence of events in the problem, (3) preserve numerical values and (4) makes the MWP solver $\mathbf{F}$ to generate an incorrect equation $\mathcal{E}^*$ for the unknown

[1]Adversarial Examples and Code is available at: https://github.com/kevivk/MWP_Adversarial

variable. We assume a black-box setting in which we have no access to the parameters, architecture or training data of the MWP solver. We only have access to the input text and equations generated by the solver.

2.2 Question Reordering

To examine whether existing MWP solvers are sensitive to the order of the question in the problem text, we moved the question Q at the start, followed by the rest of the problem description $\{S_1, S_2..S_k\}$. Formally, given the original input $\mathcal{P} = \{S_1, S_2...S_k, Q\}$ we transformed this to $\mathcal{P}^* = \{Q, S_1, S_2...S_k\}$. We keep the rest of the problem description $\{S_1, S_2..S_k\}$ unaltered. Also, to ensure that the generated problem text $\mathcal{P}^*$ is grammatically correct and fluent, we added phrases like "Given that" or "If" after the end of the question Q and before the start of the sentences $\{S_1, S_2..S_k\}$. An example of this is shown in Table 1. We additionally, make use of co-reference resolution and named entity recognition[2] to replace pronouns with their co-referent links. Note that placing the question Q at the start rather than any other position ensures that the generated problem $\mathcal{P}^*$ has the same sequence of events as the original problem $\mathcal{P}$. Moreover, this method is better than randomly shuffling the sentences in $\mathcal{P}$ as it can change the sequence of events in the problem, resulting in a completely different equation.

2.3 Sentence Paraphrasing

To check whether MWP solvers generate different equations to semantically similar inputs, we generate paraphrases of each sentence in the problem text. Sentence Paraphrasing ensures that solvers do not generate equations based on keywords and specific patterns. Formally, given a problem statement $\mathcal{P}$ we obtain top m paraphrases for each sentence S_i as $\{S_{i,1}, S_{i,2}, ..., S_{i,m}\}$ and for question Q as $\{Q_{i,1}, Q_{i,2}, ..., Q_{i,m}\}$ by passing it through a paraphrasing model $\mathcal{M}$. For sentences with numerical values present in them, we need to ensure that each paraphrase candidate associates the numeric values with the same entity and subject as it is present in the original sentence S_i. To ensure this, we follow the approach used in (Hosseini et al., 2014) to segregate each sentence S_i into entities and its subject. These are collectively labeled as head entity $h_{i,orig}$ for the original sentence S_i and $h_{i,k}$ for the paraphrase candidates $S_{i,k}$. This methodology ensures that each numeric value is still associated correctly with its attributes even after paraphrasing. Paraphrased sentences that do not have matching head entities for any of the numeric values are filtered out. The remaining paraphrases of S_i and question Q are combined to generate all possible combinations of problem texts. The input combination for which the MWP solver generates an incorrect or invalid equation is selected as the final adversarial problem text $\mathcal{P}^*$. Sentence Paraphrasing generates inputs containing small linguistic variations and diverse keywords (more examples in appendix). Therefore, it is used to evaluate whether existing MWP solvers rely on specific keywords or patterns to generate equations. Algorithm 1 shows all the steps followed above to generate paraphrases.

Algorithm 1 Sentence Paraphrasing

```
Input: Problem text $\mathcal{P}$, $\mathcal{M}$ is Paraphrase model
Output: Adversarial text $\mathcal{P}^*$
 1: $\mathcal{P}^* \leftarrow \mathcal{P}$
 2: $y_{orig} \leftarrow \mathbf{F}(\mathcal{P})$
 3: for $S_i$ in $\mathcal{P}$ do
 4:     $C \leftarrow \mathcal{M}(S_i)$
 5:     for $c_j$ in $C$ do
 6:         if $h_{i,orig} == h_{i,j}$ then
 7:             $paraphrases.add(c_j)$
 8:     $paraphrases.add(S_i)$
 9:     $candidates.add(paraphrases)$
10: for $c_k$ in $Combinations(candidates)$ do
11:     $y_{adv} \leftarrow \mathbf{F}(c_k)$
12:     if $y_{adv} \neq y_{orig}$ then
13:         $\mathcal{P}^* \leftarrow c_k$
14:         end
```

3 Experiments

3.1 Datasets and Models

We evaluate the robustness of three state-of-the-art MWP solvers: (1) *Seq2Seq* (Wang et al., 2017) having an LSTM encoder and an attention based decoder. (2) *GTS* (Xie and Sun, 2019) having an LSTM encoder and a tree based decoder and (3) *Graph2tree* (Zhang et al., 2020b) consists of a both a tree based encoder and decoder. Many existing datasets are not suitable for our analysis as either they are in Chinese (Wang et al., 2017) or they have problems of higher complexities (Huang et al., 2016) . We conduct experiments across the

[2]https://spacy.io/

two largest available English language datasets satisfying our requirements: (1) *MaWPS* (Koncel-Kedziorski et al., 2016) containing $2,373$ problems (2) *ASDIV-A* (Miao et al., 2020) containing $1,213$ problems. Both datasets have MWPs with linear equation in one variable.

3.2 Experimental Setup

We trained the three MWP solvers from scratch as implemented in baseline paper (Wang et al., 2017) on the above two datasets using 5-fold cross-validation as followed in (Zhang et al., 2020b). The original accuracies obtained on the datasets are shown in Table 2. We used (Zhang et al., 2020a) to generate paraphrases of each sentence in the problem text. Same hyperparameter values were used as present in the original implementation of the paraphrase model. We conducted a human evaluation (Section 4.3) to verify the quality of generated adversarial examples. Further details are given in Appendix.

3.3 Implementation Details

For conducting our experiments we have used two Boston SYS-7048GR-TR nodes equipped with NVIDIA GeForce GTX 1080 Ti computational GPU's . The number of parameters ranged from 20M to 130M for different models. Hyperparameter values were not modified, and we follow the recommendations of the respective models. We chose the number of candidate paraphrases m used in Algorithm 1 to be 7. Generating adversarial examples using Question Reordering took around 3 minutes on average for both MaWPS and ASDiv-A dataset. Sentence Paraphrasing took around 10 minutes on average for generation of adversarial examples on both the datasets. These experiments are not computation heavy as the generation technique is of linear order and number of examples are moderate.

3.4 Results

Table 2 shows the results of our proposed methods. On average, the generated adversarial examples can lower the accuracy of MWP solvers by over 40 percentage points. Across both datasets, *Graph2Tree*, the state-of-the-art MWP solver achieves only 34% and 24% accuracy on *Question Reordering* and *Sentence Paraphrasing* respectively. *Sentence Paraphrasing* is around 10 percentage points more successful in attacking MWP solvers than *Question Reordering*. These results verify our claim that current MWP solvers are sensitive to small variations in the input. Table 1 shows an MWP problem and its adversarial counterparts generated by our method—more examples in the Appendix section.

Dataset	**Eval Type**	**Seq2Seq**	**GTS**	**Graph2Tree**
MaWPS	Orig	53.0	82.6	83.7
	QR	**18.2**	**32.3**	**35.6**
	SP	**10.5**	**22.7**	**25.5**
ASDIV-A	Orig	54.5	71.4	77.4
	QR	**17.5**	**30.5**	**33.5**
	SP	**13.2**	**21.2**	**23.8**

Table 2: Results of MWP Solvers on adversarial examples. Orig is the original accuracy, QR is Question Reordering and SP is Sentence Paraphrasing.

4 Analysis

4.1 BERT Embeddings

We trained the solvers using pre-trained BERT embeddings and then generated adversarial examples against them using our proposed methods. Results obtained are shown in Table 3. We see that using BERT embeddings, the original accuracy of MWP solvers increases by 5 percentage points, and they are more robust than solvers trained from scratch. Specifically, these solvers do well against *Question Reordering* because of the contextualized nature of BERT embeddings, but for examples generated using *Sentence Paraphrasing* methods these models do not perform well. However, on average, our adversarial examples can lower the accuracy by 30 percentage points on both datasets.

4.2 Adversarial Training

To examine the robustness of MWP solvers against our attacks, we generated adversarial examples on the training set of both the datasets using our proposed methods and then augmented the training sets with the generated adversarial examples. We then retrained the MWP solvers and again attacked these solvers using our methods. Table 3 shows that the MWP solvers become more robust to attacks. Specifically, the solvers perform well against *Question Reordering* but are still deceived by *Sentence Paraphrasing*. Nevertheless, our proposed attack methods are still able to lower the accuracy of MWP solvers by 25 percentage points.

Dataset	Eval Type	Seq2Seq	GTS	Graph2Tree
MaWPS	Adv (QR)	32.4	52.3	54.9
	Adv (SP)	27.6	40.7	42.3
	BERT (QR)	**45.3**	**63.0**	**65.6**
	BERT (SP)	**32.5**	**43.5**	**45.5**
ASDIV-A	Adv (QR)	34.5	48.4	54.8
	Adv (SP)	28.8	31.6	33.0
	BERT (QR)	**41.3**	**59.8**	**62.7**
	BERT (SP)	**30.6**	**40.0**	**42.6**

Table 3: Accuracy of MWP solvers with adversarial training on our proposed methods. Adv and BERT represent models trained from scratch and BERT embeddings respectively.

4.3 Human Evaluation

To verify the quality and the validity of the adversarial examples, we asked human evaluators (1) To check if the paraphrases will result in the same linear equation as that of the original problem, (2) Evaluate each adversarial example in the range 0 to 1 to check its semantic similarity with the original problem and (3) On a scale of 1 to 5 rate each adversarial example for its grammatical correctness. We also explicitly check for examples which do not satisfy our evaluation criteria and manually remove them from adversarial examples set. Three different human evaluators evaluate each sample, and the mean results obtained are shown in Table 4.

Evaluation criteria	MaWPS	ASDIV-A
Same Linear Equation	85.7%	86.2%
Semantic Similarity	0.88	0.89
Grammatical Correctness	4.55	4.63

Table 4: Human Evaluation scores on datasets

5 Future Work and Conclusion

The experiments in this paper showcase that NLP models do not understand MWP entirely and are not robust enough for practical purposes. Our work encourages the development of robust MWP solvers and techniques to generate adversarial math examples. We believe that the generation of quality MWP's will immensely help develop solvers that genuinely understand numbers and text in combination. Future works could focus on creating more such techniques for adversarial examples generation and making robust MWP solvers.

6 Acknowledgments

We would like to thank the anonymous reviewers for their constructive feedback. We would also like to thank our colleagues at Data Sciences and Analytics Center, IIIT Hyderabad for providing valuable feedback. Special thanks to the human annotators who have helped in evaluation of generated adversarial examples.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998.*

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443.*

Ting-Rui Chiang and Yun-Nung Chen. 2018. Semantically-aligned equation generation for solving and reasoning math word problems. *arXiv preprint arXiv:1811.00720.*

Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv preprint arXiv:1908.06083.*

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751.*

Max Glockner, Vered Shwartz, and Yoav Goldberg. 2018. Breaking nli systems with sentences that require simple lexical inferences. *arXiv preprint arXiv:1805.02266.*

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.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328.*

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

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.

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.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2020. A context aware approach for generating natural language attacks. *arXiv preprint arXiv:2012.13339*.

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021. Generating natural language attacks in a hard label black box setting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Valencia, Spain. Association for Computational Linguistics.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*.

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.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

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.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Lei Wang, D. Zhang, Jipeng Zhang, Xing Xu, L. Gao, B. Dai, and H. Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *AAAI*.

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.

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

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. Graph-to-tree learning for solving math word problems. Association for Computational Linguistics.

Original Problem
Problem Statement : A teacher had 7 worksheets to grade . If she graded 3 , but then another 4 were turned in, how many worksheets would she have to grade ?
Predicted Equation : X = 7+3-4
Question Reordering
Problem Statement : How many worksheets would she have to grade given that a teacher had 7 worksheets to grade and if she graded 3 but then another 4 were turned in?
Predicted Equation : X = 7+3+4
Sentence Paraphrasing
Problem Statement : A teacher had her students work on 7 questions. 3 would be graded if she graded it. Then another 4 was turned in. How many things would she have to grade?
Predicted Equation : X = 7+3-4

Original Problem
Problem Statement : Gwen earned 20 points for each bag of cans she recycled . If she had 10 bags, but didn't recycle 3 of them , how many points would she have earned ?
Predicted Equation : X = (20*(10-3))
Question Reordering
Problem Statement : How many points would she have earned given that Gwen earned 20 points for each bag of cans she recycled and if she had 10 bags but didn't recycle 3 of them ?
Predicted Equation : X = 20*10-3
Sentence Paraphrasing
Problem Statement : Gwen earned 20 points for each bag of cans she recycled. She have 10 bags. She did not recycle 3 of them. How many points would she have gotten?
Predicted Equation : X = 20+10-3

Original Problem
Problem Statement : : Dennis has 12 pencils stored in boxes. If there are 3 boxes, how many pencils must go in each box?
Predicted Equation : X = 12/3
Question Reordering
Problem Statement : : If there are 3 boxes, how many pencils must go in each box given that Dennis has 12 pencils stored in boxes ?
Predicted Equation : X = 12/3
Sentence Paraphrasing
Problem Statement : Dennis has 12 pencils in boxes. There are 3 boxes. Find the number of pencils in each box?
Predicted Equation : X = 12-3

Original Problem
Problem Statement : Oliver made 10 dollars mowing lawns over the summer . If he spent 4 dollars buying new mower blades. How many 3 dollar games could he buy with the money he had left ?
Predicted Equation : X = (10-4)/3
Question Reordering
Problem Statement : How many 3 dollar games could Oliver buy with the money he had left given that Oliver made 10 dollars mowing lawns over the summer and if he spent 4 dollars buying new mower blades.
Predicted Equation : X = (10-4)*3
Sentence Paraphrasing
Problem Statement : Over the summer, Oliver made 10 dollars mowing lawns. He spent 4 dollars on new blades. With the money he had left, how many 3 dollar games could he buy?
Predicted Equation : X = (10-4)*3

Table 5: Some instances of valid Adversarial Examples

Original Problem
Problem Statement : A trivia team had 10 members total. But during a game 2 members did not show up. If each member that did show up scored 3 points. How many points were scored?
Predicted Equation : X = (10-2)*3
Sentence Paraphrasing
Problem Statement : A team with 10 members had a lot of questions to answer. But during the game 2 members did not show up. 3 points were scored if each member showed up. How many points were scored?

Original Problem
Problem Statement : A tailor cut 15 of an inch off a skirt and 5 of an inch off a pair of pants . How much more did the tailor cut off the skirt than the pants ?
Predicted Equation : X = 15-5
Sentence Paraphrasing
Problem Statement : The 15 was cut by a tailor. There is a skirt and 5 of an inch off. There is a pair of pants. How much more did the tailor cut off the skirt than the pants?

Original Problem
Problem Statement : A vase can hold 10 flowers . If you had 5 carnations and 5 roses,
how many vases would you need to hold the flowers?
Predicted Equation : X = (5+5)/10
Sentence Paraphrasing
Problem Statement : 10 flowers can be held in a vase. If you had 5 and 5 roses. How many vases
do you need to hold the flowers.

Table 6: Some instances of invalid Adversarial Examples