Are NLP Models really able to Solve Simple Math Word Problems?

Arkil Patel **Satwik Bhattamishra** **Navin Goyal**
Microsoft Research India
{t-arkpat,t-satbh,navingo}@microsoft.com

Abstract

The problem of designing NLP solvers for math word problems (MWP) has seen sustained research activity and steady gains in the test accuracy. Since existing solvers achieve high performance on the benchmark datasets for elementary level MWPs containing one-unknown arithmetic word problems, such problems are often considered "solved" with the bulk of research attention moving to more complex MWPs. In this paper, we restrict our attention to English MWPs taught in grades four and lower. We provide strong evidence that the existing MWP solvers rely on shallow heuristics to achieve high performance on the benchmark datasets. To this end, we show that MWP solvers that do not have access to the question asked in the MWP can still solve a large fraction of MWPs. Similarly, models that treat MWPs as bag-of-words can also achieve surprisingly high accuracy. Further, we introduce a challenge dataset, SVAMP, created by applying carefully chosen variations over examples sampled from existing datasets. The best accuracy achieved by state-of-the-art models is substantially lower on SVAMP, thus showing that much remains to be done even for the simplest of the MWPs.

1 Introduction

A Math Word Problem (MWP) consists of a short natural language narrative describing a state of the world and poses a question about some unknown quantities (see Table 1 for some examples). MWPs are taught in primary and higher schools. The MWP task is a type of semantic parsing task where given an MWP the goal is to generate an expression (more generally, equations), which can then be evaluated to get the answer. The task is challenging because a machine needs to extract relevant information from natural language text as well as perform mathematical reasoning to solve it. The complexity of MWPs can be measured along multiple axes, e.g., reasoning and linguistic complexity and world and domain knowledge. A combined complexity measure is the *grade level* of an MWP, which is the grade in which similar MWPs are taught. Over the past few decades many approaches have been developed to solve MWPs with significant activity in the last decade (Zhang et al., 2020).

PROBLEM: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Jack have now? 8 - 3 = 5
QUESTION SENSITIVITY VARIATION: Jack had 8 pens and Mary had 5 pens. Jack gave 3 pens to Mary. How many pens does Mary have now? 5 + 3 = 8
REASONING ABILITY VARIATION: Jack had 8 pens and Mary had 5 pens. Mary gave 3 pens to Jack. How many pens does Jack have now? 8 + 3 = 11
STRUCTURAL INVARIANCE VARIATION: Jack gave 3 pens to Mary. If Jack had 8 pens and Mary had 5 pens initially, how many pens does Jack have now? 8 - 3 = 5

Table 1: Example of a Math Word Problem along with the types of variations that we make to create SVAMP.

MWPs come in many varieties. Among the simplest are the one-unknown arithmetic word problems where the output is a mathematical expression involving numbers and one or more arithmetic operators $(+, -, *, /)$. (Problems in Tables 1 and 6 are of this type.) More complex MWPs may have systems of equations as output or involve other operators or may involve more advanced topics and specialized knowledge. Recently, researchers have started focusing on solving such MWPs, e.g. multiple-unknown linear word problems (Huang et al., 2016a), geometry (Sachan and Xing, 2017) and probability (Amini et al., 2019), believing that existing work can handle one-unknown arithmetic MWPs well (Qin et al., 2020). In this paper, we question the capabilities of the state-of-the-art

(SOTA) methods to robustly solve even the simplest of MWPs suggesting that the above belief is not well-founded.

In this paper, we provide concrete evidence to show that existing methods use shallow heuristics to solve a majority of word problems in the benchmark datasets. We find that existing models are able to achieve reasonably high accuracy on MWPs from which the question text has been removed leaving only the narrative describing the state of the world. This indicates that the models can rely on superficial patterns present in the narrative of the MWP and achieve high accuracy without even looking at the question. In addition, we show that a model without the word-order information (i.e., the model treats the MWP as a bag-of-words) can also solve the majority of the MWPs in the benchmark datasets.

The presence of these issues in existing benchmarks makes them unreliable for measuring the performance of models. Hence, we create a challenge set called SVAMP (**S**imple **V**ariations on **A**rithmetic **M**ath word **P**roblems; pronounced *swamp*) of one-unknown arithmetic word problems with grade level up to 4 by applying simple variations over word problems in an existing dataset (see Table 1 for some examples). SVAMP further highlights the brittle nature of existing models when trained on these benchmark datasets. On evaluating SOTA models on SVAMP, we find that they are not even able to solve half the problems in the dataset. This failure of SOTA models on SVAMP points to the extent to which they rely on simple heuristics in training data to make prediction.

Below, we summarize the two broad contributions of our paper.

- We show that the majority of problems in benchmark datasets can be solved by shallow heuristics lacking word-order information or lacking question text.
- We create a challenge set called SVAMP for more robust evaluation of methods developed to solve elementary level math word problems.

2 Related Work

Math Word Problems. A wide variety of methods and datasets have been proposed to solve MWPs; e.g. statistical machine learning (Roy and Roth, 2018), semantic parsing (Huang et al., 2017) and most recently deep learning (Wang et al., 2017; Xie and Sun, 2019; Zhang et al., 2020); see (Zhang et al., 2020) for an extensive survey. Many papers have pointed out various deficiencies with previous datasets and proposed new ones to address them. Koncel-Kedziorski et al. (2016) curated the MAWPS dataset from previous datasets which along with Math23k (Wang et al., 2017) has been used as benchmark in recent works. Recently, ASDiv (Miao et al., 2020) has been proposed to provide more diverse problems with annotations for equation, problem type and grade level. HMWP (Qin et al., 2020) is another newly proposed dataset of Chinese MWPs that includes examples with muliple-unknown variables and requiring non-linear equations to solve them.

Identifying artifacts in datasets has been done for the Natural Language Inference (NLI) task by McCoy et al. (2019), Poliak et al. (2018), and Gururangan et al. (2018). Rosenman et al. (2020) identified shallow heuristics in a Relation Extraction dataset. Cai et al. (2017) showed that biases prevalent in the ROC stories cloze task allowed models to yield state-of-the-art results when trained only on the endings. To the best of our knowledge, this kind of analysis has not been done on any Math Word Problem dataset.

Challenge Sets for NLP tasks have been proposed most notably for NLI and machine translation (Belinkov and Glass, 2019; Nie et al., 2020; Ribeiro et al., 2020). Gardner et al. (2020) suggested creating *contrast sets* by manually perturbing test instances in small yet meaningful ways that change the gold label. We believe that we are the first to introduce a challenge set targeted specifically for robust evaluation of Math Word Problems.

3 Background

3.1 Problem Formulation

We denote a Math Word Problem P by a sequence of n tokens $P = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n)$ where each token $\boldsymbol{w}_i$ can be either a word from a natural language or a numerical value. The word problem P can be broken down into *body* $B = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k)$ and *question* $Q = (\boldsymbol{w}_{k+1}, \ldots, \boldsymbol{w}_n)$. The goal is to map P to a valid mathematical expression E_P composed of numbers from P and mathematical operators from the set $\{+, -, /, *\}$ (e.g. $3 + 5 - 4$). The metric used to evaluate models on the MWP task is Execution Accuracy, which is obtained from comparing the predicted answer (calculated by evalu-

Model	MAWPS	ASDiv-A
Seq2Seq (S)	79.7	55.5
Seq2Seq (R)	86.7	76.9
GTS (S) (Xie and Sun, 2019)	82.6	71.4
GTS (R)	88.5	81.2
Graph2Tree (S) (Zhang et al., 2020)	83.7	77.4
Graph2Tree (R)	**88.7**	**82.2**
Majority Template Baseline	17.7	21.2

Table 2: 5-fold Cross Validation Accuracies (↑) of baseline models on datasets. (R) means that the model is provided with RoBERTa pretrained embeddings while (S) means that the model is trained from scratch.

ating E_P) with the annotated answer. In this work, we focus on single-unknown arithmetic word problems.

3.2 Datasets and Methods

Many of the existing datasets are not suitable for our analysis as either they are in Chinese, e.g. Math23k (Wang et al., 2017) and HMWP (Qin et al., 2020), or have harder problem types, e.g. Dolphin18K (Huang et al., 2016b). We consider the widely used benchmark MAWPS (Koncel-Kedziorski et al., 2016) composed of 2373 MWPs and the arithmetic subset of ASDiv (Miao et al., 2020) called ASDiv-A which has 1218 MWPs mostly up to grade level 4 (MAWPS does not have grade level information). Both MAWPS and ASDiv-A are evaluated on 5-fold cross validation based on pre-assigned splits.
We consider three models in our experiments:
(a) Seq2Seq consists of a Bidirectional LSTM Encoder to encode the input sequence and an LSTM decoder with attention (Luong et al., 2015) to generate the equation.
(b) GTS (Xie and Sun, 2019) uses an LSTM Encoder to encode the input sequence and a tree-based Decoder to generate the equation.
(c) Graph2Tree (Zhang et al., 2020) combines a Graph-based Encoder with a Tree-based Decoder.

The performance of these models on both datasets is shown in Table 2. We either provide RoBERTa (Liu et al., 2019) pre-trained embeddings to the models or train them from scratch. Graph2Tree (Zhang et al., 2020) with RoBERTa embeddings achieves the state-of-the-art for both datasets. Note that our implementations achieve a higher score than the previously reported highest score of 78% on ASDiv-A (Miao et al., 2020) and 83.7% on MAWPS (Zhang et al., 2020). The implementation details are provided in Section A in the Appendix.

Model	MAWPS	ASDiv-A
Seq2Seq	77.4	58.7
GTS	76.2	60.7
Graph2Tree	**77.7**	**64.4**

Table 3: 5-fold Cross Validation Accuracies (↑) of baseline models on Question-removed datasets

4 Deficiencies in existing datasets

Here we describe the experiments showing that there are important deficiencies in MAWPS and ASDiv-A.

4.1 Evaluation on Question-removed MWPs

As mentioned in Section 3.1, each MWP consists of a body B, which provides a short narrative on a state of the world and a question Q, which inquires about an unknown quantity about the state of the world. For each fold in the provided 5-fold split in MAWPS and ASDiv-A, we keep the train set unchanged while we remove the questions Q from the problems in the test set. Hence, each problem in the test set consists of only the body B without any question Q. We evaluate all three models with RoBERTa embeddings on these datasets. The results are provided in Table 3.

The best performing model is able to achieve a 5-fold accuracy of 64.4% on ASDiv-A and 77.7% on MAWPS. Loosely translated, this means that nearly 64% of the problems in ASDiv-A and 78% of the problems in MAWPS can be correctly answered without even looking at the question. This suggests the presence of patterns in the bodies of MWPs in these datasets with a direct correlation with the output equation.

Some recent works have also demonstrated similar evidence of bias in NLI datasets (Gururangan et al., 2018; Poliak et al., 2018). They observed that NLI models were able to predict the correct label for a large fraction of the standard NLI datasets based on only the hypothesis of the input and without the premise. Our results on question-removed examples of math word problems resembles their observations on NLI datasets and similarly indicates the presence of artifacts that help statistical models predict the correct answer without complete information. Note that even though the two methods appear similar, there is an important distinction. In Gururangan et al. (2018), the model is

Model	MAWPS		ASDiv-A	
	Easy	*Hard*	*Easy*	*Hard*
Seq2Seq	86.8	86.7	91.3	56.1
GTS	92.6	71.7	91.6	65.3
Graph2Tree	93.4	71.0	92.8	63.3

Table 4: Results of baseline models on the *Easy* and *Hard* test sets.

Model	MAWPS	ASDiv-A
FFN + LSTM Decoder (S)	75.1	46.3
FFN + LSTM Decoder (R)	**77.9**	**51.2**

Table 5: 5-fold Cross Validation Accuracies (↑) of the constrained model on the datasets. (R) denotes that the model is provided with non-contextual RoBERTa pre-trained embeddings while (S) denotes that the model is trained from scratch.

trained and tested on hypothesis only examples and hence, the model is forced to find artifacts in the hypothesis during *training*. On the other hand, our setting is more natural since the model is trained in the standard way on examples with both the body and the question. Thus, the model is not explicitly forced to learn based on the body during training and our results not only show the presence of artifacts in the datasets but also suggest that the SOTA models exploit them.

Following Gururangan et al. (2018), we attempt to understand the extent to which SOTA models rely on the presence of simple heuristics in the body to predict correctly. We partition the test set into two subsets for each model: problems that the model predicted correctly without the question are labeled *Easy* and the problems that the model could not answer correctly without the question are labeled *Hard*. We carry out this experiment for all the three models.

Table 4 shows the performance of each model on their respective *Hard* and *Easy* sets. Note that their performance on the full set is already provided in Table 2. It can be seen clearly that although the models correctly answer many *Hard* problems, the bulk of their success is attributed to the *Easy* problems. This shows that the ability of SOTA methods to robustly solve word problems is overestimated and that they can rely on simple heuristics in the body of the problems to make their prediction.

4.2 Performance of a constrained model

We construct a simple model based on the Seq2Seq architecture by removing the LSTM Encoder and replacing it with a Feed-Forward Network that maps the input embeddings to their hidden representations. The LSTM Decoder is provided with the **average** of these hidden representations as its initial hidden state. During decoding, an attention mechanism (Luong et al., 2015) assigns weights to individual hidden representations of the input tokens. We use either RoBERTa embeddings (non-contextual; taken directly from the Embedding Matrix) or train the model from scratch. Clearly, this model cannot use word-order information.

Table 5 shows the performance of this model on MAWPS and ASDiv-A. The constrained model with non-contextual RoBERTa embeddings is able to achieve a cross-validation accuracy of 51.2 on ASDiv-A and an astounding 77.9 on MAWPS. It is surprising to see that a model having no word-order information can solve a majority of word problems in these datasets.

These results indicate that it is possible to get a good score on these datasets by simply associating the occurence of specific words in the word problems to their corresponding equations. We illustrate this more clearly in the next section.

4.3 Analyzing the attention weights

To get a better understanding of how the constrained model is able to perform so well, we analyze the attention weights that it assigns to the hidden representations of the input tokens. As shown by Wiegreffe and Pinter (2019), analyzing the attention weights of our constrained model is a reliable way to explain its prediction since each hidden representation consists of information about only that token as opposed to the case of an RNN where each hidden representation may have information about the context i.e. its neighboring tokens.

We train the contrained model (with RoBERTa embeddings) on the full ASDiv-A dataset and observe the attention weights it assigns to the words of the input problems. We found that the model usually attends to a single word to make its prediction, irrespective of the context. Table 6 shows some representative examples. In the first example, the model assigns an attention weight of 1 to the representation of the word *'every'* and predicts the correct equation. However, when we make a subtle change to this problem such that the corresponding equation changes, the model keeps on attending over the word *'every'* and predicts the same equation, which is now incorrect. Similar observations

Input Problem	Predicted Equation	Answer
John delivered 3 letters at every house. If he delivered for 8 houses, how many letters did John deliver?	3 * 8	24 ✓
John delivered 3 letters at every house. He delivered 24 letters in all. How many houses did John visit to deliver letters?	3 * 24	72 ✗
Sam made 8 dollars mowing lawns over the Summer. He charged 2 bucks for each lawn. How many lawns did he mow?	8 / 2	4 ✓
Sam mowed 4 lawns over the Summer. If he charged 2 bucks for each lawn, how much did he earn?	4 / 2	2 ✗
10 apples were in the box. 6 are red and the rest are green. how many green apples are in the box?	10 - 6	4 ✓
10 apples were in the box. Each apple is either red or green. 6 apples are red. how many green apples are in the box?	10 / 6	1.67 ✗

Table 6: Attention paid to specific words by the constrained model.

can be made for the other two examples. Table 22 in the appendix has more such examples.

Note that, we do not claim that every model trained on these datasets relies on the occurrence of specific words in the input problem for prediction the way our constrained model does. We are only asserting that it is possible to achieve a good score on these datasets even with such a brittle model, which clearly makes these datasets unreliable to robustly measure model performance.

5 SVAMP

The efficacy of existing models on benchmark datasets has led to a shift in the focus of researchers towards more difficult MWPs. We claim that this efficacy on benchmarks is misleading and SOTA MWP solvers are unable to solve even elementary level one-unknown MWPs. To this end, we create a challenge set named SVAMP containing simple one-unknown arithmetic word problems of grade level up to 4. The examples in SVAMP test a model across different aspects of solving word problems. For instance, a model needs to be sensitive to questions and possess certain reasoning abilities to correctly solve the examples in our challenge set. SVAMP is similar in terms of scope and difficulty for humans to existing datasets of the same level, but is less susceptible to being solved by models relying on superficial patterns.

Our work differs from adversarial data collection methods such as Adversarial NLI (Nie et al., 2020) in that these methods create examples depending on the failure of a particular model while we create examples without referring to any specific model. Inspired by the notion of *Normative evaluation* (Linzen, 2020), our goal is to create a dataset of simple problems that any system designed to solve MWPs should be expected to solve. We create new problems by applying certain variations to existing problems, similar to the work of Ribeiro et al. (2020). However, unlike their work, our variations do not check for linguistic capabilities. Rather, the choice of our variations is motivated by the experiments in Section 4 as well as certain simple capabilities that any MWP solver must possess.

5.1 Creating SVAMP

We create SVAMP by applying certain types of variations to a set of seed examples sampled from the ASDiv-A dataset. We select the seed examples from the recently proposed ASDiv-A dataset since it appears to be of higher quality and harder than the MAWPS dataset: We perform a simple experiment to test the coverage of each dataset by training a model on one dataset and testing it on the other one. For instance, when we train a Graph2Tree model on ASDiv-A, it achieves 82% accuracy on MAWPS. However, when trained on MAWPS and tested on ASDiv-A, the model achieved only 73% accuracy. Also recall Table 2 where most models performed better on MAWPS. Moreover, ASDiv has problems annotated according to types and grade levels which are useful for us.

To select a subset of seed examples that sufficiently represent different types of problems in the ASDiv-A dataset, we first divide the examples into groups according to their annotated types. We discard types such as *'TVQ-Change'*, *'TVQ-Initial'*, *'Ceil-Division'* and *'Floor-Division'* that have less than 20 examples each. We also do not consider the *'Difference'* type since it requires the use of an additional modulus operator. For ease of vari-

Group	Examples in ASDiv-A	Selected Seed Examples
Addition	278	28
Subtraction	362	33
Multiplication	188	19
Division	176	20
Total	1004	100

Table 7: Distribution of selected seed examples across types.

CATEGORY	VARIATION	EXAMPLES
Question Sensitivity	Same Object, Different Structure	**Original:** Allan brought two balloons and Jake brought four balloons to the park. How many balloons did Allan and Jake have in the park? **Variation:** Allan brought two balloons and Jake brought four balloons to the park. How many more balloons did Jake have than Allan in the park?
	Different Object, Same Structure	**Original:** In a school, there are 542 girls and 387 boys. 290 more boys joined the school. How many pupils are in the school? **Variation:** In a school, there are 542 girls and 387 boys. 290 more boys joined the school. How many boys are in the school?
	Different Object, Different Structure	**Original:** He then went to see the oranges being harvested. He found out that they harvest 83 sacks per day and that each sack contains 12 oranges. How many sacks of oranges will they have after 6 days of harvest? **Variation:** He then went to see the oranges being harvested. He found out that they harvest 83 sacks per day and that each sack contains 12 oranges. How many oranges do they harvest per day?
Reasoning Ability	Add relevant information	**Original:** Every day, Ryan spends 4 hours on learning English and 3 hours on learning Chinese. How many hours does he spend on learning English and Chinese in all? **Variation:** Every day, Ryan spends 4 hours on learning English and 3 hours on learning Chinese. If he learns for 3 days, how many hours does he spend on learning English and Chinese in all?
	Change Information	**Original:** Jack had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now? **Variation:** Dorothy had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Dorothy have now?
	Invert Operation	**Original:** He also made some juice from fresh oranges. If he used 2 oranges per glass of juice and he made 6 glasses of juice, how many oranges did he use? **Variation:** He also made some juice from fresh oranges. If he used 2 oranges per glass of juice and he used up 12 oranges, how many glasses of juice did he make?
Structural Invariance	Change order of objects	**Original:** John has 8 marbles and 3 stones. How many more marbles than stones does he have? **Variation:** John has 3 stones and 8 marbles. How many more marbles than stones does he have?
	Change order of phrases	**Original:** Matthew had 27 crackers. If Matthew gave equal numbers of crackers to his 9 friends, how many crackers did each person eat? **Variation:** Matthew gave equal numbers of crackers to his 9 friends. If Matthew had a total of 27 crackers initially, how many crackers did each person eat?
	Add irrelevant information	**Original:** Jack had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now? **Variation:** Jack had 142 pencils. Dorothy had 50 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now?

Table 8: Types of Variations with examples. '**Original:**' denotes the base example from which the variation is created, '**Variation:**' denotes a manually created variation.

ation creation, we discard the few examples that are more than 40 words in length. To control the complexity of resulting variations, we only consider those problems as seed examples which can be solved by an expression with a single operator. Then, within each group, we cluster examples using K-Means algorithm over RoBERTa sentence embeddings of each example. From each cluster, the example closest to the cluster centroid is selected as a seed example. We select a total of 100 seed examples in this manner. The distribution of seed examples according to different types of problems can be seen in Table 7.

5.1.1 Variations

The variations that we make to each seed example can be broadly classified into three categories based on desirable properties of an ideal model: *Question Sensitivity*, *Reasoning Ability* and *Structural Invariance*. Examples of each type of variation are provided in Table 8.

1. Question Sensitivity. Variations in this category check if the model's answer depends on the question. In these variations, we change the question in the seed example while keeping the body the same. The possible variations are as follows:
(a) Same Object, Different Structure: The principal object (i.e. object whose quantity is unknown) in the question is kept the same while the structure of the question is changed.
(b) Different Object, Same Structure: The principal object in the question is changed while the structure of question remains fixed.
(c) Different Object, Different Structure: Both, the principal object in the question and the structure of the question, are changed.

2. Reasoning Ability. Variations here check whether a model has the ability to correctly determine a change in reasoning arising from subtle changes in the problem text. The different possible variations are as follows:
(a) Add relevant information: Extra relevant in-

Dataset	# Problems	# Equation Templates	# Avg Ops	CLD
MAWPS	2373	39	1.78	0.26
ASDiv-A	1218	19	1.23	0.50
SVAMP	1000	26	1.24	0.22

Table 9: Statistics of our dataset compared with MAWPS and ASDiv-A.

formation is added to the example that affects the output equation.
(b) Change information: The information provided in the example is changed.
(c) Invert operation: The previously unknown quantity is now provided as information and the question instead asks about a previously known quantity which is now unknown.

3. Structural Invariance. Variations in this category check whether a model remains invariant to superficial structural changes that do not alter the answer or the reasoning required to solve the example. The different possible variations are as follows:
(a) Add irrelevant information: Extra irrelevant information is added to the problem text that is not required to solve the example.
(b) Change order of objects: The order of objects appearing in the example is changed.
(c) Change order of phrases: The order of number-containing phrases appearing in the example is changed.

5.1.2 Protocol for creating variations

Since creating variations requires a high level of familiarity with the task, the construction of SVAMP is done in-house by the authors and colleagues, hereafter called the *workers*. The 100 seed examples (as shown in Table 7) are distributed among the *workers*.

For each seed example, the *worker* needs to create new variations by applying the variation types discussed in Sec. 5.1.1. Importantly, a combination of different variations over the seed example can also be done. For each new example created, the worker needs to annotate it with the equation as well as the type of variation(s) used to create it. More details about the creation protocol can be found in Appendix B.

We created a total of 1098 examples. However, since ASDiv-A does not have examples with equations of more than two operators, we discarded 98 examples from our set which had equations consisting of more than two operators. This is to ensure that our challenge set does not have any unfairly difficult examples. The final set of 1000 examples was provided to an external volunteer unfamiliar with the task to check the grammatical and logical correctness of each example.

5.2 Dataset Properties

Our challenge set SVAMP consists of one-unknown arithmetic word problems which can be solved by expressions requiring no more than two operators. Table 9 shows some statistics of our dataset and of ASDiv-A and MAWPS. The Equation Template for each example is obtained by converting the corresponding equation into prefix form and masking out all numbers with a meta symbol. Observe that the number of distinct Equation Templates and the Average Number of Operators are similar for SVAMP and ASDiv-A and are considerably smaller than for MAWPS. This indicates that SVAMP does not contain unfairly difficult MWPs in terms of the arithmetic expression expected to be produced by a model.

Previous works, including those introducing MAWPS and ASDiv, have tried to capture the notion of *diversity* in MWP datasets. Miao et al. (2020) introduced a metric called Corpus Lexicon Diversity (CLD) to measure lexical diversity. Their contention was that higher lexical diversity is correlated with the quality of a dataset. As can be seen from Table 9, SVAMP has a much lesser CLD than ASDiv-A. SVAMP is also less diverse in terms of problem types compared to ASDiv-a. Despite this we will show in the next section that SVAMP is in fact more challenging than ASDiv-A for current models. Thus, we believe that lexical diversity is not a reliable way to measure the quality of MWP datasets. Rather it could depend on other factors such as the diversity in MWP structure which preclude models exploiting shallow heuristics.

5.3 Experiments on SVAMP

We train the three considered models on a combination of MAWPS and ASDiv-A and test them on SVAMP. The scores of all three models with and without RoBERTa embeddings for various subsets of SVAMP can be seen in Table 10.

The best performing Graph2Tree model is only able to achieve an accuracy of 43.8% on SVAMP. This indicates that the problems in SVAMP are indeed more challenging for the models than the problems in ASDiv-A and MAWPS despite being

	Seq2Seq		GTS		Graph2Tree	
	S	*R*	*S*	*R*	*S*	*R*
Full Set	24.2	40.3	30.8	41.0	36.5	**43.8**
One-Op	25.4	42.6	31.7	44.6	42.9	**51.9**
Two-Op	20.3	**33.1**	27.9	29.7	16.1	17.8
ADD	28.5	**41.9**	35.8	36.3	24.9	36.8
SUB	22.3	35.1	26.7	36.9	41.3	**41.3**
MUL	17.9	**38.7**	29.2	**38.7**	27.4	35.8
DIV	29.3	56.3	39.5	61.1	40.7	**65.3**

Table 10: Results of models on the SVAMP challenge set. *S* indicates that the model is trained from scratch. *R* indicates that the model was trained with RoBERTa embeddings. The first row shows the results for the full dataset. The next two rows show the results for subsets of SVAMP composed of examples that have equations with one operator and two operators respectively. The last four rows show the results for subsets of SVAMP composed of examples of type Addition, Subtraction, Multiplication and Division respectively.

of the same scope and type and less diverse. Table 23 in the Appendix lists some simple examples from SVAMP on which the best performing model fails. This indicates that the existing models cannot robustly solve elementary level word problems.

The high performance of Graph2Tree on non-commutative operators is likely due to the fact that it has a dedicated *'Quantity Comparison Graph'* in its encoder to predict the order of numbers around the operator as per their magnitude. It is perhaps surprising to see that the Seq2Seq baseline performs better than the SOTA models on Two-Operator, Addition and Multiplication problems.

Next, we remove the questions from the examples in SVAMP and evaluate them using the three models with RoBERTa embeddings trained on combined MAWPS and ASDiv-A. The scores can be seen in Table 11. The accuracy drops by half when compared to ASDiv-A and more than half compared to MAWPS suggesting that the problems in SVAMP are more sensitive to the information present in the question. We also evaluate the performance of the constrained model on SVAMP

Model	SVAMP w/o ques	ASDiv-A w/o ques
Seq2Seq	29.2	58.7
GTS	28.6	60.7
Graph2Tree	**30.8**	**64.4**

Table 11: Accuracies (↑) of models on SVAMP without questions. The 5-fold CV accuracy scores for ASDiv-A without questions are restated for easier comparison.

Model	SVAMP
FFN + LSTM Decoder (S)	17.5
FFN + LSTM Decoder (R)	**18.3**
Majority Template Baseline	11.7

Table 12: Accuracies (↑) of the constrained model on SVAMP. (R) denotes that the model is provided with non-contextual RoBERTa pretrained embeddings while (S) denotes that the model is trained from scratch. The Majority Template Baseline score is the accuracy on the dataset when the model always predicts the equation template with the highest frequency.

when trained on MAWPS and ASDiv-A. The best performing model achieves only 18.3% accuracy which is marginally better than the majority template baseline. This shows that the problems in SVAMP are less vulnerable to being solved by models using simple patterns and that a model needs contextual information in order to solve them.

We also explored using SVAMP for training by combining it with ASDiv-A and MAWPS. We performed 5-fold cross validation over SVAMP where the model was trained on a combination of the three datasets and tested on unseen examples from SVAMP. To create the folds, we first divide the seed examples into five sets, with each *type* of example distributed nearly equally among the sets. A fold is obtained by combining all the examples in SVAMP that were created using the seed examples in a set. In this way, we get five different folds from the five sets. We found that the best model achieved about 65% accuracy. This indicates that even with additional training data existing models are still not close to the performance that was estimated based on prior benchmark datasets.

To check the influence of different categories of variations in SVAMP, for each category, we measure the difference between the accuracy of the

Removed Category	# Removed Examples	Change in Accuracy (Δ)
Question Sensitivity	462	-13.7
Reasoning Ability	649	+3.3
Structural Invariance	467	-4.5

Table 13: Change in accuracies when categories are removed. The Change in Accuracy $\Delta = Acc(Full) - Acc(Full - Cat)$, where $Acc(Full)$ is the accuracy on the full set and $Acc(Full - Cat)$ is the accuracy on the set of examples left after removing all examples which were created using Category Cat either by itself, or in use with other categories.

Removed Variation	# Removed Examples	Change in Accuracy (Δ)
Same Obj, Diff Struct	325	-7.3
Diff Obj, Same Struct	69	-1.5
Diff Obj, Diff Struct	74	-1.3
Add Rel Info	264	-5.5
Change Info	149	-3.2
Invert Operation	255	+10.2
Change order of Obj	107	-2.3
Change order of Phrases	152	+3.3
Add Irrel Info	281	-6.9

Table 14: Change in accuracies when categories are removed. The Change in Accuracy $\Delta = Acc(Full) - Acc(Full - Var)$, where $Acc(Full)$ is the accuracy on the full set and $Acc(Full - Var)$ is the accuracy on the set of examples left after removing all examples which were created using Variation Var either by itself, or in use with other variations.

best model on the full dataset and its accuracy on a subset containing no example created from that category of variations. The results are shown in Table 13. Both the *Question Sensitivity* and *Structural Invariance* categories of variations make SVAMP more challenging. The increase in accuracy for the *Reasoning Ability* category of variations can be attributed in large part to the *Invert Operation* type of variations. This is not surprising because most of the examples created from the *Invert Operation* variation are almost indistinguishable from normal examples in ASDiv-A which the model has seen during training. The scores for each individual variation are provided in Table 14.

We also check the break-up of performance of the best performing Graph2Tree (Zhang et al., 2020) model according to the number of numbers present in the text of the input problem. We trained the model on both ASDiv-A and MAWPS and tested on SVAMP and compare those results against the 5-fold Cross Validation setting of ASDiv-A. The scores are provided in Table 15. While the model can solve many problems consisting of only two numbers in the input text (even in our challenge set), it performs very badly on problems having more than two numbers. This shows that current methods are incapable of properly associating numbers to their context. Also, the gap between the performance on ASDiv-A and SVAMP is high, indicating that the examples in SVAMP are more difficult for these models to solve than the examples in ASDiv-A even when considering the structurally same type of Math Word Problems.

Dataset	2 nums	3 nums	4 nums
ASDiv-A	93.3	59.0	47.5
SVAMP	78.3	25.4	25.4

Table 15: Accuracy break-up according to the number of numbers in the input problem. **2 nums** refers to the subset of problems which have only 2 numbers in the problem text. Similarly, **3 nums** and **4 nums** are subsets that contain 3 and 4 different numbers in the problem text respectively.

6 Final Remarks

Going back to the original question, are existing NLP models able to solve elementary math word problems? This paper gives a negative answer. We have empirically shown that the benchmark English MWP datasets suffer from artifacts making them unreliable to gauge the performance of MWP solvers: we demonstrated that the majority of problems in the existing datasets can be solved by simple heuristics even without word-order information or the question text.

The performance of the existing models in our proposed challenge dataset also highlights their limitations in solving simple elementary level word problems. We hope that our challenge set SVAMP containing elementary level MWPs will enable more robust evaluation of methods. We believe that methods proposed in the future that make genuine advances in solving the task rather than relying on simple heuristics will perform well on SVAMP despite being trained on other datasets such as ASDiv-A and MAWPS.

In recent years, the focus of the community has shifted towards solving more difficult MWPs such as non-linear equations and word problems with multiple unknown variables. We demonstrated that the capability of existing models to solve simple one-unknown arithmetic word problems is overestimated. We believe that developing more robust methods for solving elementary MWPs remains a significant open problem.

Acknowledgements

We thank the anonymous reviewers for their constructive comments. We would also like to thank our colleagues at Microsoft Research for providing valuable feedback. We thank Monojit Choudhury for discussions about creating the dataset. We also thank Vageesh Chandramouli and Nalin Patel for their help in dataset construction.

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.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Zheng Cai, Lifu Tu, and Kevin Gimpel. 2017. Pay attention to the ending:strong neural baselines for the ROC story cloze task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 616–622, Vancouver, Canada. Association for Computational Linguistics.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814, Copenhagen, Denmark. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016a. 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, Berlin, Germany. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016b. 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, Berlin, Germany. Association for Computational Linguistics.

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.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

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

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

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

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018.

Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Shachar Rosenman, Alon Jacovi, and Yoav Goldberg. 2020. Exposing Shallow Heuristics of Relation Extraction Models with Challenge Data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3702–3710, Online. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics*, 6:159–172.

Mrinmaya Sachan and Eric Xing. 2017. Learning to solve geometry problems from natural language demonstrations in textbooks. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 251–261, Vancouver, Canada. Association for Computational Linguistics.

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.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China. Association for Computational Linguistics.

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

D. Zhang, L. Wang, L. Zhang, B. T. Dai, and H. T. Shen. 2020. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2287–2305.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937, Online. Association for Computational Linguistics.

A Implementation Details

We use 8 NVIDIA Tesla P100 GPUs each with 16 GB memory to run our experiments.The hyperparameters used for each model are shown in Table 16. The best hyperparameters are highlighted in bold. Following the setting of Zhang et al. (2020), the arithmetic word problems from MAWPS are divided into five folds, each of equal test size. For ASDiv-A, we consider the 5-fold split [238, 238, 238, 238, 266] provided by the authors (Miao et al., 2020).

B Creation Protocol

We create variations in template form. Generating more data by scaling up from these templates or by performing automatic operations on these templates is left for future work. The template form of an example is created by replacing certain words with their respective tags. Table 17 lists the various tags used in the templates.

The $[NUM]$ tag is used to replace all the numbers and the $[NAME]$ tag is used to replace all the Names of Persons in the example. The $[OBJs]$ and $[OBJp]$ tags are used for replacing the objects in the example. The $[OBJs]$ and $[OBJp]$ tags with the same index represent the same object in singular and plural form respectively. The intention when using the $[OBJs]$ or the $[OBJp]$ tag is that it can be used as a placeholder for other similar words, which when entered in that place, make sense as per the context. These tags must not be used for collectives; rather they should be used for the things that the collective represents. Some example uses of $[OBJs]$ and $[OBJp]$ tags are provided in Table 18. Lastly, the $[MOD]$ tag must be used to replace any modifier preceding the $[OBJs]$/$[OBJp]$ tag.

A preprocessing script is executed over the ***Seed Examples*** to automatically generate template suggestions for the *workers*. The script uses Named Entity Recognition and Regular Expression matching to automatically mask the names of persons and the numbers found in the ***Seed Examples***. The outputs from the script are called the ***Script Examples***. An illustration is provided in Table 19.

Each *worker* is provided with the ***Seed Examples*** along with their respective ***Script Examples*** that have been alloted to them. The *worker's* task is to edit the ***Script Example*** by correcting any mistake made by the preprocessing script and adding any new tags such as the $[OBJs]$ and the $[OBJp]$ tags in order to create the ***Base Example***. If a *worker* introduces a new tag, they need to mark it against its example-specific value. If the tag is used to mask objects, the *worker* needs to mark both the singular and plural form of the object in a comma-seperated manner. Additionally, for each unique index of $[OBJs]$/$[OBJp]$ tag in the example, the *worker* must enter atleast one alternate value that can be used in that place. Similarly, the *worker* must enter atleast two modifier words that can be used to precede the principal $[OBJs]$/$[OBJp]$ tags in the example. These alternate values are used to gather a lexicon which can be utilised to scale-up the data at a later stage. An illustration of this process is provided in Table 20.

In order to create the variations, the *worker* needs to check the different types of variations in Table 8 to see if they can be applied to the ***Base Example***. If applicable, the *worker* needs to create the ***Variation Example*** while also making a note of the type of variation. If a particular example is the result of performing multiple types of variations, all types of variations should be listed according to their order of application from latest to earliest in a comma-seperated manner. For any variation, if a *worker* introduces a new tag, they need to mark it against its example-specific value as mentioned before. The index of any new tag introduced needs to be one more than the highest index already in use for that tag in the ***Base Example*** or its previously created variations.

To make the annotation more efficient and streamlined, we provide the following steps to be followed in order:

1. Apply the *Question Sensitivity* variations on the ***Base Example***.

2. Apply the *Invert Operation* variation on the ***Base Example*** and on all the variations obtained so far.

3. Apply the *Add relevant information* variation on the ***Base Example***. Then considering these variations as ***Base Examples***, apply the *Question Sensitivity* variations.

4. Apply the *Add irrelevant information* variation on the ***Base Example*** and on all the variations obtained so far.

5. Apply the *Change information* variation on the ***Base Example*** and on all the variations obtained so far.

	Seq2Seq		**GTS**		**Graph2Tree**		**Constrained**	
Hyperparameters	Scratch	RoBERTa	Scratch	RoBERTa	Scratch	RoBERTa	Scratch	RoBERTa
Embedding Size	[128, **256**]	[768]	[**128**, 256]	[768]	[**128**, 256]	[768]	[**128**, 256]	[768]
Hidden Size	[**256**, 384]	[**256**, 384]	[384, **512**]	[384, **512**]	[256, **384**]	[256, **384**]	[256, **384**]	[**256**, 384]
Number of Layers	[1, **2**]	[1, **2**]	[1, **2**]	[1, **2**]	[1, **2**]	[1, **2**]	[**1**, 2]	[**1**, 2]
Learning Rate	[5e-4, 8e-4, **1e-3**]	[1e-4, **2e-4**, 5e-4]	[8e-4, 1e-3, **2e-3**]	[5e-4, 8e-4, **1e-3**]	[**8e-4**, 1e-3, 2e-3]	[5e-4, **8e-4**, 1e-3]	[**1e-3**, 2e-3]	[1e-3, **2e-3**]
Embedding LR	[5e-4, **8e-4**, 1e-3]	[5e-6, **8e-6**, 1e-5]	[8e-4, 1e-3, **2e-3**]	[5e-6, **8e-6**, 1e-5]	[**8e-4**, 1e-3, 2e-3]	[5e-6, 8e-6, **1e-5**]	[**1e-3**, 2e-3]	[1e-3, **2e-3**]
Batch Size	[**8**, 16]	[**4**, 8]	[8, **16**]	[**4**, 8]	[**8**, 16]	[4, **8**]	[8, **16**]	[**4**, 8]
Dropout	[0.1]	[0.1]	[0.5]	[0.5]	[0.5]	[0.5]	[0.1]	[0.1]
# Parameters	8.5M	130M	15M	140M	16M	143M	5M	130M
Epochs	60	50	60	50	60	50	60	50
Avg Time/Epoch	10	40	60	120	60	120	10	15

Table 16: Different hyperparameters and the values considered for each of them in the models. The best hyperparameters for each model for 5-fold Cross Validation on ASDiv-A are highlighted in bold. Average Time/Epoch is measured in seconds.

Tag	**Description**
[NUM*x*]	Number
[NAME*x*]	Names of Persons
[OBJs*x*]	Singular Object
[OBJp*x*]	Plural Object
[MOD*x*]	Modifier

Table 17: List of tags used in annotated templates. ***x*** denotes the index of the tag.

6. Apply the *Change order of Objects* and *Change order of Events or Phrases* variations on the ***Base Example*** and on all the variations obtained so far.

Table 21 provides some variations for the example in Table 20. Note that two seperate examples were created through the *'Add irrelevant information'* variation. The first by applying the variation on the ***Original Example*** and the second by applying it on a previously created example (as directed in Step-4).

To make sure that different *workers* following our protocol make similar types of variations, we hold a trial where each worker created variations from the same 5 seed examples. We observed that barring minor linguistic differences, most of the created examples were the same, thereby indicating the effectiveness of our protocol.

C Analyzing Attention Weights

In Table 22, we provide more examples to illustrate the specific word to equation correlation that the constrained model learns.

D Examples of Simple Problems

In Table 23, we provide a few simple examples from SVAMP that the best performing Graph2Tree model could not solve.

E Ethical Considerations

In this paper, we consider the task of automatically solving Math Word Problems (MWPs). Our work encourages the development of better systems that can robustly solve MWPs. Such systems can be deployed for use in the education domain. E.g., an application can be developed that takes MWPs as input and provides detailed explanations to solve them. Such applications can aide elementary school students in learning and practicing math.

We present a challenge set called SVAMP of one-unknown English Math Word Problems. SVAMP is created in-house by the authors themselves by applying some simple variations to examples from ASDiv-A (Miao et al., 2020), which is a publicly available dataset. We provide a detailed creation protocol in the Appendix section B. We are not aware of any risks associated with our proposed dataset.

To provide an estimate of the energy requirements of our experiments, we provide the details such as computing platform and running time in Appendix section A. Also, in order to reduce carbon costs from our experiments, we first perform a broad hyperparameter search over only a single fold for the datasets and then run the cross validation experiment over a select few hyperparameters.

Excerpt of Example	Beth has 4 packs of red crayons and 2 packs of green crayons. Each pack has 10 crayons in it.
Template Form	$[NAME1]$ has $[NUM1]$ packs of $[MOD1]$ $[OBJp1]$ and $[NUM2]$ packs of $[MOD2]$ $[OBJp1]$.
Excerpt of Example	In a game, Frank defeated 6 enemies. Each enemy earned him 9 points.
Template Form	In a game $[NAME1]$ defeated $[NUM1]$ $[OBJp1]$. Each $[OBJs1]$ earned him $[NUM2]$ points.

Table 18: Example uses of tags. Note that in the first example, the word *'packs'* was not replaced since it is a collective. In the second example, the word *'points'* was not replaced because it is too instance-specific and no other word can be used in that place.

Seed Example Body	Beth has 4 packs of crayons. Each pack has 10 crayons in it. She also has 6 extra crayons.
Seed Example Question	How many crayons does Beth have altogether?
Seed Example Equation	4*10+6
Script Example Body	$[NAME1]$ has $[NUM1]$ packs of crayons . Each pack has $[NUM2]$ crayons in it . She also has $[NUM3]$ extra crayons .
Script Example Question	How many crayons does $[NAME1]$ have altogether ?
Script Example Equation	$[NUM1] * [NUM2] + [NUM3]$

Table 19: An example of suggested templates. Note that the preprocessing script could not succesfully tag *crayons* as $[OBJp1]$.

Script Example Body	$[NAME1]$ has $[NUM1]$ packs of crayons . Each pack has $[NUM2]$ crayons in it . She also has $[NUM3]$ extra crayons .
Script Example Question	How many crayons does $[NAME1]$ have altogether ?
Base Example Body	$[NAME1]$ has $[NUM1]$ packs of $[OBJp1]$. Each pack has $[NUM2]$ $[OBJp1]$ in it . She also has $[NUM3]$ extra $[OBJp1]$.
Base Example Question	How many $[OBJp1]$ does $[NAME1]$ have altogether ?
$[OBJ1]$	crayon, crayons
Alternate for $[OBJ1]$	pencil, pencils
Alternate for $[MOD]$	small, large

Table 20: An example of editing the Suggested Templates. The edits are indicated in green.

Base Example Body	$[NAME1]$ has $[NUM1]$ packs of $[OBJp1]$. Each pack has $[NUM2]$ $[OBJp1]$ in it. She also has $[NUM3]$ extra $[OBJp1]$
Base Example Question	How many $[OBJp1]$ does $[NAME1]$ have altogether ?
Base Example Equation	$[NUM1] * [NUM2] + [NUM3]$
Category	Question Sensitivity
Variation	Same Object, Different Structure
Variation Body	$[NAME1]$ has $[NUM1]$ packs of $[OBJp1]$. Each pack has $[NUM2]$ $[OBJp1]$ in it. She also has $[NUM3]$ extra $[OBJp1]$.
Variation Question	How many $[OBJp1]$ does $[NAME1]$ have in packs?
Variation Equation	$[NUM1] * [NUM2]$
Category	Structural Invariance
Variation	Add irrelevant information
Variation Body	$[NAME1]$ has $[NUM1]$ packs of $[OBJp1]$ and $[NUM4]$ packs of $[OBJp2]$. Each pack has $[NUM2]$ $[OBJp1]$ in it. She also has $[NUM3]$ extra $[OBJp1]$.
Variation Question	How many $[OBJp1]$ does $[NAME1]$ have altogether ?
Variation Equation	$[NUM1] * [NUM2] + [NUM3]$
Variation Body	$[NAME1]$ has $[NUM1]$ packs of $[OBJp1]$ and $[NUM4]$ packs of $[OBJp2]$. Each pack has $[NUM2]$ $[OBJp1]$ in it. She also has $[NUM3]$ extra $[OBJp1]$.
Variation Question	How many $[OBJp1]$ does $[NAME1]$ have in packs?
Variation Equation	$[NUM1] * [NUM2]$

Table 21: Example Variations

Input Problem	**Predicted Equation**	**Answer**
Mike had 8 games. After he gave some to his friend he had 5 left . How many games did he give to his friend?	8 - 5	3 ✓
After Mike gave some games to his friend he had 5 left . If he had 8 games initially, how many games did he give to his friend?	5 - 8	-3 ✗
Jack bought 5 radios but only 2 of them worked. How many radios did not work?	5 - 2	3 ✓
Jack bought 5 radios but only 2 of them worked. How many more radios did not work than those that did?	5 - 2	3 ✗
Ross had 6 marbles. He sold 2 marbles to Joey. How many marbles does Ross have now?	6 - 2	4 ✓
Ross had 6 marbles. Joey sold 2 marbles to Ross. How many marbles does Ross have now?	6 - 2	4 ✗
Bob collected 7 cans. He lost 3 of them. How many cans does Bob have now?	7 - 3	4 ✓
Bob had 7 cans. He collected 3 more. How many cans does Bob have now?	7 - 3	4 ✗
Joey had 9 pens. he used 4 of them. How many pens does he have now?	9 - 4	5 ✓
Joey used 4 pens. If he had 9 pens intially, how many pens does he have now?	4 - 9	-5 ✗
Jill read 30 pages in 10 days. How many pages did she read per day?	30 / 10	3 ✓
Jill can read 3 pages per day. How many pages can she read in 10 days?	3 / 10	0.33 ✗
Mary's hair was 15 inches long. After she did a haircut, it was 10 inches long . how much did she cut off ?	15 - 10	5 ✓
Mary cut off 5 inches of her hair. If her hair is now 10 inches long, how long was it earlier?	5 - 10	-5 ✗

Table 22: Attention paid to specific words by the constrained model.

Input Problem	**Correct Equation**	**Predicted Equation**
Every day ryan spends 6 hours on learning english and 2 hours on learning chinese. How many more hours does he spend on learning english than he does on learning chinese?	6 - 2	2 - 6
In a school there are 34 girls and 841 boys. How many more boys than girls does the school have?	841 - 34	34 - 841
David did 44 push-ups in gym class today. David did 9 more push-ups than zachary. How many push-ups did zachary do?	44 - 9	44 + 9
Dan has $ 3 left with him after he bought a candy bar for $ 2. How much money did he have initially?	3 + 2	3 - 2
Jake has 11 fewer peaches than steven. If jake has 17 peaches. How many peaches does steven have?	11 + 17	17 - 11
Kelly gives away 91 nintendo games. How many did she have initially if she still has 92 games left?	91 + 92	92 - 91
Emily is making bead necklaces for her friends. She was able to make 18 necklaces and she had 6 beads. How many beads did each necklace need?	18 / 6	6 / 18
Frank was reading through some books. Each book had 249 pages and it took frank 3 days to finish each book. How many pages did he read per day?	249 / 3	(249 * 3) / 3
A mailman has to give 5 pieces of junk mail to each block. If he gives 25 mails to each house in a block, how many houses are there in a block?	25 / 5	5 / 25
Faye was placing her pencils and crayons into 19 rows with 4 pencils and 27 crayons in each row. How many pencils does she have?	19 * 4	19 * 27
White t - shirts can be purchased in packages of 53. If mom buys 57 packages of white t - shirts and 34 trousers, How many white t - shirts will she have?	53 * 57	(53 * 57) + 34
An industrial machine can make 6 shirts a minute. It worked for 5 minutes yesterday and for 12 minutes today. How many shirts did machine make today?	6 * 12	5 + 12

Table 23: Some simple examples from SVAMP on which the best performing Graph2Tree model fails.