Evaluating and Enhancing the Robustness of Math Word Problem Solvers

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Vivek Kumar
2019701004
vivek.k@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
June 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **'Evaluating and Enhancing the Robustness of Math Word Problem Solvers**' by Vivek Kumar, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. Vikram Pudi

To my dearest parents and beloved sister
for their boundless love and support

Acknowledgments

I am truly thankful for the opportunity to pursue my masters at IIIT Hyderabad. The journey here has been filled with a perfect mix of research experiences, wonderful memories, and friendships that will last a lifetime. It has been a truly unique and once-in-a-lifetime opportunity.

I am deeply grateful to my advisor Dr. Vikram Pudi for his invaluable guidance and support throughout my research. His unwavering encouragement, even when the results were not immediately clear, allowed me to continue pushing forward. He provided me with ample support and complete autonomy to pursue the problems that interested me, and he made time in his busy schedule to meet with me regularly to discuss the progress of my work and potential future directions. I am also thankful for his willingness to allow me to collaborate with others and for giving me the freedom to work at my own pace. Without his support and guidance, this thesis would not have been possible.

I would like to specifically thank Rishabh for validating my ideas, providing valuable insights and suggestions. He took the time to carefully analyze my research problem and provide me with perspectives that helped to shape my work.

I would also like to acknowledge the role of my friends Debtanu, Gaurav, Kuntal, Pranav, Shubham Mante, Shubham Raj and Tushar Abhishek for making my time at IIIT memorable. From our enlightening discussions about research to random topics, to exploring Hyderabad and navigating the uncertain times of the COVID-19 pandemic, we made countless memories and supported each other along the way.

I am grateful to IIIT Hyderabad for providing me with an excellent environment for conducting my research. The courses, assignments, and seminar talks offered numerous opportunities for students to explore their research interests. In today's world, access to reliable computational infrastructure is essential for conducting research in computer science, especially when working with large data sets. I am fortunate to have had access to a wealth of computational resources at IIIT Hyderabad, which allowed me to carry out my experiments successfully. I am deeply indebted to the institute for providing such a conducive environment for my research.

Lastly, but most importantly, I would like to acknowledge the endless support and sacrifices made by my parents, without their blessings none of this would have been possible. They have been my pillars of strength and have always supported me in creating my own path. I would also like to extend a very special acknowledgement to my sister, Pratibha, whose unwavering support and belief in me were instrumental in making this work possible.

Abstract

In the recent past, math word problem solvers have received wide attention from the NLP community at large. With the advancement of deep learning techniques, solvers have started to show better performance than traditional rule based semantic parsing techniques. Standard accuracy metrics have shown that math word problem solvers have achieved high performance on benchmark datasets. However, these performances are based on datasets that have limited problem statements and equation templates, thus providing very limited diversity and a low probability of generalization on different word problems for practical purposes. Hence, the extent to which existing MWP solvers truly understand natural language problem statements and its relationship with numerical quantities is still unclear. In this work, we first 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. These results showcase that math word solvers do not generalize well and rely on superficial cues to achieve high performance.

Next, we conduct experiments to showcase that this behaviour is mainly associated with the limited size and diversity present in existing MWP datasets. We modify the problem statements by altering the text in different settings such that either the problem statement does not make much sense or no question has been asked in the problem statement. The preliminary results from these analysis did not show significant drop in accuracy metric as was expected. Then, we propose several data augmentation techniques broadly categorized into Substitution and Paraphrasing based methods to mitigate the issues found by our analysis. By deploying data augmentation methods we increase the size of existing datasets by five folds. Extensive experiments on two benchmark datasets across three state-of-the-art MWP solvers show that the proposed methods increase the generalization and robustness of existing solvers. On average, the proposed methods significantly increase the state-of-the-art results by over five percentage points on benchmark datasets. Further, the solvers trained on the augmented dataset performs comparatively better on the challenge test set. We also show the effectiveness of proposed techniques through ablation studies and verify the quality of augmented samples through human evaluation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Developing mathematical reasoning in machines has been a long standing problem. This problem has been considered as the key to solve artificial general intelligence. Early works which aim to solve this task can be traced back to 1960s [7] [8] [17] . Mathematical understanding has always been seen as a necessary skill which machines should have in order to execute other complex tasks such as robotic manipulations, self driving cars, human-like chat bots among others. Particularly, having the ability to understand numerical quantities along with world knowledge is an essential and very desired skill. This is extremely challenging because we need to make the machines understand the world-view and let them infer the numerical quantities with their legitimate mathematical operations. The recent advances in the field of natural language processing can be attributed to the introduction of deep learning and availability of large amounts of text data with enhanced computing infrastructure that has enabled machines to understand natural language well. AI systems have been able to execute complex NLP tasks like machine translation, question answering, dialogue systems and paraphrase generation among others with a fair degree of effectiveness.

Original Problem
Problem: 348 teddy bears are sold for Rs 23 each. There are total 470 teddy bears in a store and the remaining teddy bears are sold for Rs. 17 each. How much did the store earn after selling all the teddy bears ?
Solution Equation: $x = 348 * 23 + (470 - 348) * 17$

Table 1.1: A sample math word problem statement

1.2 Problem Description

Solving math world problems which are school level algebra problems is a natural extension to extend the capabilities and understanding of NLP systems. These problems have a natural language text which narrates a world view with numerical quantities. These world views are narrated to perform mathematical manipulations on the quantities. Later, a question is asked to find out the value of some unknown quantities. Table 1.1 shows a sample math word problem statement. Here, after describing a narrative involving numerical quantities and certain transactions involving them, a question is asked about finding out the value of one of the quantities. A solution expression is the answer that a solver must generate. This expression when solved for the unknown variable should return the correct value for the desired quantity.

Recently, math word problem solvers is gaining a lot of attention. Many solvers have been proposed that attempt to tackle this challenging problem [72][77][82]. They show high performance on the limited size labelled datasets that are available (Question - Equation pair). Gathering large number of high quality Question - Equation pairs with relevant meta-data is in itself a huge annotation task. However, it is still not understood what are the factors that affect the performance of math solvers. It is important to address this question as the dataset size is limited and we want to make sure that the solvers are able to generate new equations, solve other categories of problems and generalize well on the problem solving task. In this work, we propose to probe the robustness of the existing solvers by generating adversarial math problems and test the performance of existing solvers on these perturbed problem statements. We further, showcase techniques using which we can increase the robustness of existing solvers.

1.3 Thesis Contribution

Following are the key contributions made in this work:

- 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.

- This is the first work that extensively evaluates data augmentation techniques for MWP solving. This is the first attempt to generate MWP problems automatically without manual intervention.

- On average, the generated adversarial examples are able to reduce the accuracy of MWP solvers by over 40 percent points. 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 are valid, semantically similar and grammatically correct.

- Accuracy of the state of the art solvers increases after training on the proposed augmented dataset. This demonstrates the effectiveness of our methods. To verify the validity of generated augmentations we conduct human evaluation studies.
- We increase the diversity of the training dataset through augmentations and obtain comparatively better results than state-of-the art solvers on the SVAMP [52] challenge set.

1.4 Thesis Layout

The thesis is organized in the following manner:

- Chapter 2 gives a detailed overview about the background and recent trends in Math Word Problems. We discuss in detail three of the baseline papers, their architectural choices, their merits and demerits. Further, we also take a deep dive to understand the background and related work in data augmentation techniques for text and adversarial attacks for text inputs.
- In Chapter 3, we introduce our first agenda, that is to probe existing solvers by generating adversarial examples. We take a deep look at the novel strategies used to generate adversarial examples by overcoming the challenges posed by the task setup. We further perform human validation and other experiments to showcase the effectiveness of our proposed method.
- In Chapter 4, we identify the issues that cause low robustness in solvers. We propose an hypothesis and attempt to prove it empirically through a set of well designed experiments. Further, we propose set of data augmentation techniques to enhance the diversity and count of training samples in existing datasets.
- Finally, we conclude our thesis with an overview of the work that we have done and the possible future works that can enhance the performance of solvers.

Chapter 2

Background and Related Works

2.1 Math Word Problem Solvers

In this section, we will go through the background and related works for math word problem solvers. We will go through the key ideas, challenges involved and discuss about three key baseline papers that have shaped the area of math word problem solving.

2.1.1 Deep Neural Solver for Math Word Problems

Motivation: This work [72] introduced the use of deep neural networks for solving math word problems automatically compared to previously described statistical approaches. In this work, the authors attempt to map math word problems to equation templates using a recurrent neural network based model. The authors also implement a similarity-based retrieval model and compare its performance with the sequence to sequence based model. The observations show that sequence based models outperforms retrieval based models on an average. However, the retrieval model is able to correctly answer many questions for which sequence model produces incorrect results. The findings also show that the accuracy of the similarity based retrieval model correlates positively with the maximal similarity score between the problems in training data and the target question. It implies that higher the similarity score, more is the accuracy.

Based on their observations, the authors proposed a hybrid model which combines both the sequence based and retrieval based models. In the hybrid model, the retrieval model is selected only if the similarity score returned by the retrieval model is greater than a set threshold, else sequence based model is chosen to solve the problem. To facilitate the development of future math word solvers, a large dataset in Chinese language is developed. This dataset Math23K [72] comprises of over $23,000$ math word problems in one variable.

Methodology:

Number Mapping: For a math word problem P which has m known numbers, a number mapping M_p maps the numbers in the problem statement P against numerical token identifiers $\{n_1, ..., n_m\}$ as per their occurrence in the problem statement.

Template Creation: A template represents a general form of an equation where true numbers are represented by their respective numerical variables. For any problem statement P with equation E_p and number mapping list M_p, its equation template is derived by mapping numbers in the solution equation E_p to a list of number token identifiers $\{n_1, ..., n_m\}$ according to M_p. Let's understand both number mapping and template creation through an example illustrated in Table 2.1.

Problem Statement:
Text: Dan has 5 pens and 3 pencils, Jessica have 4 more pens and 2 less pencils than him. How many pens and pencils does Jessica have in total ?
Solution Equation:
Equation: $X = 5 + 4 + 3 - 2$
Number Mapping
$M : \{n_1 = 5, n_2 = 3, n_3 = 4, n_4 = 2\}$
Equation Template
$X = n_1 + n_3 + n_2 - n_4$

Table 2.1: A MWP problem and the process of number mapping and equation template creation.

2.1.1.1 RNN Based Sequence to Sequence Model

Significant Number Identification (SNI): In a math word problem, not all the numbers present in the problem statement are required in the equation for solving the problem. An example is shown in Table, where the numbers "1" in "1 day, 1 girl" and number "2" in "She has 2 types of" should not be used in equation construction. A number is assumed to be significant if the number should be included in the equation to solve the problem; otherwise it is insignificant. For the problem in Table 4, significant numbers are 9, 3, and 5, while 1 and 2 are insignificant numbers. Identifying significant and insignificant numbers is important for constructing correct equations. For this purpose, the authors have proposed to build a LSTM-based binary classification model to determine whether a number in a problem text is significant or insignificant.

1. The model architecture for significant number identification is showcased in Figure 2.1. The key idea is to leverage the sequential capability of LSTMs and perform a forward over the problem text to generate a representation of the text. The authors view this problem as a binary classification problem. The training data for SNI model is extracted from the math word problems. Each number and its context in problems is a training instance of SNI. Each problem text is divided into chunks of text

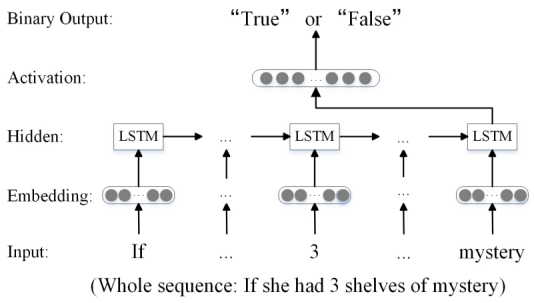

Figure 2.1: The significant number identification model architecture.

having a numerical quantity and the labels "True" or "False" are derived form the solution equation of the training data. If the numerical quantity present in the chunk is used in the solution equation then they label it as true else they label it as false. The performance of SNI is very high with the accuracy reported around 99.3%.

2. A sequence to sequence model with GRU based encoder and LSTM based decoder is chosen as the model architecture. The activation function at decoder side has been modified with the rules to adapt to math word problem. Following modifications were incorporated:

- Rule 1: If r_{t-1} in $\{+, -, *, /\}$, then r_t will not be in $\{+, -, *, /,), =\}$;
- Rule 2: If r_{t-1} is a number, then r_t will not be a number and not in $\{(, =\}$;
- Rule 3: If r_{t-1} is " $=$ ", then r_t will be not in $\{+, -, *, /, =,)\}$;
- Rule 4: If r_{t-1} is "(", then r_t will not be in $\{(,), +, -, *, /, =\}$;
- Rule 5: If r_{t-1} is ")", then r_t will not be a number and not in $\{(,)\}$;

Directly generating equation templates by a soft-max function can lead to the generation of some erroneous equations, such as: $'x = n1 + +n2'$ and $'x = (n1 * n2'$. To make sure that the final output equations are mathematically correct, the authors need to enforce the above rules which basically constraint the generation of certain characters based on previously generated characters.

2.1.1.2 Retrieval Model

Figure 2.3 showcases the percentage distribution of problems solved by retrieval model and seq2seq model. As we can see retrieval model augments the type of problems correctly solved by seq2seq model. Hence, their ensemble is desired and will result in overall higher performance. The retrieval model attempts to solve the math problems by calculating the lexical similarity between the test problem against each problem in the training data. The equation template of the most similar problem is

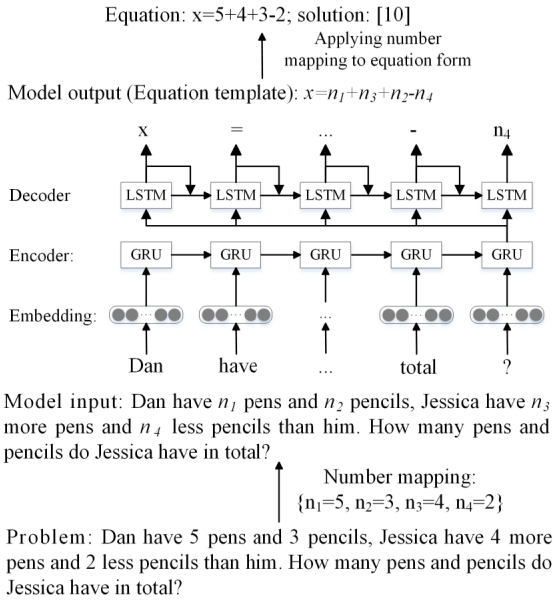

Figure 2.2: Seq2Seq architecture of Deep Neural Solver.

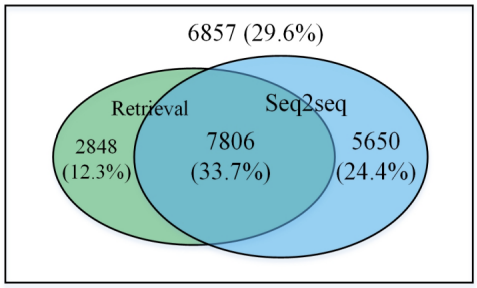

Figure 2.3: Correctly solved problems distribution between retrieval and Seq2Seq models.

Algorithm 1 Hybrid model

Input: Q: problems in training data;
P_T: testing problem;
θ: pre-defined threshold of similarity

Output: Problem solution

1: Get equation templates and number mappings for training problems Q and testing problem P_T.
2: Number identification: identify significant numbers
3: Retrieval:
choose problem Q_1 from Q that has the maximal Jaccard similarity with P_T
4: **if** $J(P_T, Q_1) > \theta$ **then**
5: Apply the retrieval model: select equation template T of Q_1
6: **else**
7: Apply the seq2seq model: $T = seq2seq(P_T)$
8: **end if**
9: Applying number mappings of P_T to T and calculating final solution

Figure 2.4: Step by step algorithm of hybrid model.

then retrieved and applied to the test problem statement. Jaccard similarity is used to compute the similarity between the test problem and train equation templates.The overall algorithm of hybrid model is showcased in figure 2.4.

2.1.1.3 Results

Figure 2.5 showcases the accuracy results of the Seq2Seq model, retrieval model and hybrid model with and without SNI. These results testify the effectiveness of SNI and hybrid model. Overall, with the help of results and experiments the authors conclude that with larger training data they witness an increase in model accuracy, the hybrid model improves performance by over 22 percent on the baselines and have the capability to generate new templates apart from those already present in the training set.

2.1.2 GTS : A Goal-Driven Tree-Structured Neural Model for Math Word Problems

Motivation: This work [77] is an improvement over the RNN based sequence to sequence method for solving MWP. It attempts to conceptualize human behaviour while solving math word problems. When

	Math23K	Alg514
ZDC	42.1%	**79.7%**
Retrieval model w/o SNI	46.0%	70.1%
Retrieval model w/ SNI	47.2%	70.1%
Seq2seq model w/o SNI	53.7%	17.2%
Seq2seq model w/ SNI	58.1%	16.1%
Hybrid model w/o SNI	61.1%	70.1%
Hybrid model w/ SNI	**64.7%**	70.1%

Figure 2.5: Results of hybrid model in different settings on Math23k and Alg514 datasets.

a human reads the problem statement of a MWP, they first figure out which target quantity needs to be set as the goal, and after that they start paying attention to the relevant information present in the problem statement. If the goal can be achieved by the relevant information, the problem solving has been completed; otherwise, human needs to further decompose the goal into sub-goals and combine them together by a required operator to solve the MWP. The major issue with Seq2Seq-based models is that they may generate invalid expressions which are mathematically incorrect. For instance, it is difficult to determine exactly how many consecutive left parentheses would be required before the inside operations are handled, this may sometimes lead to the wrong expressions.

Inspired by the goal-driven mechanism in human math problem solving, the author's create a model to generate an expression tree from the given problem statement. The generated expression tree follows a top-down goal decomposition method, the model initializes the root goal vector which represents the ultimate goal of the problem, and then a context vector is utilized to summarize the relevant information from the problem statement. Using the goal vector and its context vector, a token is predicted which implicitly helps in deciding whether the goal should be broken down into further sub goals or not. If the predicted token is a numeric value or constant quantity, the goal is realized directly; otherwise (i.e., the predicted token is an operator), two new sub-goal vectors (one for left sub-goal and the other for the right) will be generated. Similarly, the prediction and goal decomposition process are repeated for them. However, for a commutative operator such as "+" or "×", its right sub-goal may be the same as the left one, due to its commutative property. To address this issue, the model completes the construction of the left sub-tree before generating the right sub-goal. The generation of right sub-goal takes the information of its left sibling sub-tree into consideration, which is encoded as a sub-tree embedding by a recursive neural network.

Methodology:
The overall methodology is explained below:

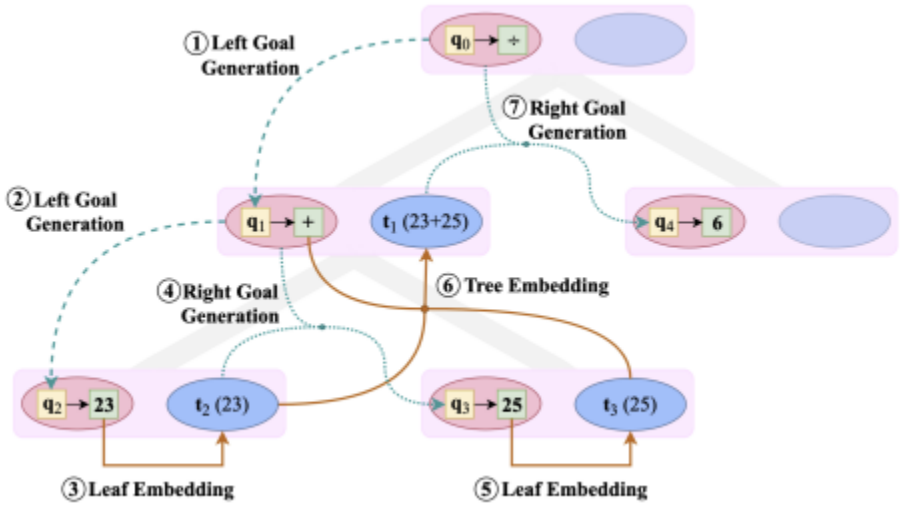

Figure 2.6: GTS Goal Decomposition Workflow.

- **Goal Decomposition:** The goal decomposition method workflow is depicted in the figure 2.6. Each node n present in the expression tree T consists of the following components: a goal vector q_n, a token $\hat{y}$, and a sub-tree embedding t of node n. The goal vector q_n is used for instructing how to construct the sub-tree root from node n, and the generated sub-tree is then used to realize the goal. To do so, the method first predicts the token $\hat{y}$ from the goal vector q_n. In turn, the predicted token determines whether the goal should be broken down into further sub-goals or not. If the predicted token is a mathematical operator, the goal will be further decomposed into two sub-goals, a left sub-goal q_l and a right sub-goal q_r. The left sub-goal along with the sub-tree embedding is utilized to construct the right sub-tree of n. If the predicted token is a constant or a numerical quantity the goal will be simply realized by the predicted token. This goal decomposition process is conducted recursively similar to the depth-first traversal.

- **Sub-Goal Generation:** Before understanding left and right sub-goal generation we must first understand how a context vector is computed and then used to calculate the predicted tokens. Given the input query, we first extract the embeddings of each token using a trainable matrix representation. These embeddings are then passed down through a GRU based bi-directional encoder to compute the hidden states corresponding to each token position. The bidirectional outputs at each position are summed up to obtain the absolute hidden states. The context vector c is a weighted summation over these hidden states. The weights are calculated as attention weights between the goal vector q_n and position wise hidden state h_p. Output token $\hat{y}$ is computed as the arg-max of the probability distribution vector obtained from soft-max in the vocabulary dimension space. We transfer the context vector and goal vector in the vocabulary space using a feed-forward fully connected layer.

Model	Accuracy(%)
Hybrid model w/ SNI [Wang *et al.*, 2017]	64.7
Ensemble model w/ EN [Wang *et al.*, 2018a]	68.4
GTS model w/o Subtree Embedding	70.0
GTS model	**74.3**

Figure 2.7: Accuracy comparison of different models with GTS.

Case 1: The store shipped in a batch of leather shoes. NUM(n_0 [$\frac{1}{3}$]) of the total was sold on the first day, and NUM(n_1 [$\frac{3}{5}$]) of the first day's sale was sold on the second day. There were NUM(n_2 [280]) pairs left. How many pairs of leather shoes did the store bring in?
Seq2Seq: $\div n_2 - 1 n_0 * n_0 n_1$;**(error)** **GTS:** $\div n_2 - -1 n_0 * n_0 n_1$;**(correct)**

Case 2: Of the NUM(n_0 [697]) combined shipment equipments of Shenzhou NUM(n_1 [7]) spacecraft, NUM(n_2 [346]) are followed, NUM(n_3 [237]) are updated, and the rest are newly developed. How many new equipments are there?
Seq2Seq: $- - n_0 n_3 n_4$;**(error)** **GTS:** $- - n_0 n_2 n_3$;**(correct)**

Case 3: Guangming Primary School spent NUM(n_0 [288]) yuan on NUM(n_1 [12]) chairs. And then NUM(n_2 [36]) chairs of the same kind were bought. How much did the school spend on chairs?
GTS w/o Subtree Embedding: $\times \div n_0 n_1 \div n_0 n_1$;**(error)** **GTS:** $\times \div n_0 n_1 + n_1 n_2$;**(correct)**

Figure 2.8: Some MWP examples showcasing effectiveness of GTS.

After computation of the context vector c, the left sub-tree node q_l is computed on the basis of context vector c and output token $\hat{y}$, with gating functions and a $2-$layer feed forward neural network.The right child q_r takes into account the left child sub-tree q_l, which has been generated prior to the right child due to pre-order traversal. The left sub-tree is encoded bottom up as t_l according to the recursive neural network. Then the right goal vector q_r of the right child is calculated as a combination of the gating functions and feed-forward neural networks.

Results:
Figure 2.7 reports the answer accuracies to compare and highlight the effectiveness of goal decomposition and sub-tree methods. Figure 2.8 showcases examples illustrating the effectiveness of GTS over Seq2Seq models. However, despite promising results over Seq2Seq models, GTS suffers when the tree expression lengths or solution equation length is larger. This is intuitive as the complexity of solving the expression tree will increase with expression length.

2.1.3 Graph2Tree: Graph-to-Tree Learning for Solving Math Word Problems

Motivation: With the introduction of GTS, it became evident that tree-based neural decoder models have a positive impact on the performance of existing math word solvers, however, most of these models are unable to represent the order and relationship information between the quantities present in the problem statement. This results in poor quantity representations and generation of incorrect and ambiguous solution expressions. Graph2Tree [82] attempts to combine the advantages of a graph based encoder and a tree based decoder to improve the performance of math word solvers. They introduce two

Case 1: The class organized students to climb the mountain. The female students were divided into 4 groups, and each group had 15 students. There were 76 male students in total. How many students joined climbing last week?
GTS: $(15 + 76) * 4$; **(error)** **Graph2Tree**: $15 * 4 + 76$;
Case 2: Lingling and Yaya are 200 meters apart. Lingling is in the front and runs 3 meters per second. Yaya is in the rear and runs 5 meters per second. They set off at the same time, running in the same direction. How long will it be before Yaya could catch up with Lingling?
GTS: $200/(3 - 5)$; **(error)** **Graph2Tree**: $200/(5 - 3)$;
Case 3: A bus and a truck departed from the two cities of A and B, which are 900 kilometers apart. They went in opposite directions. It takes 10 hours for the bus to travel from A to B, and 15 hours for the truck to travel from B to A. How many hours would it be before the bus, and the truck meet?
GTS: $900/(900/10 + 1/15)$; **(error)** **Graph2Tree**: $900/(900/10 + 900/15)$;

Figure 2.9: Examples of solving math word problems with Graph2Tree model.

graph quantities namely *quantity cell graph* and *quantity comparison graph*. These graphs are designed to mitigate the limitations and drawbacks of existing solvers by effectively representing the order and relationship information between the different quantities present in the problem statement.

To enhance the overall representation of a quantity, the relationship between the textual content associated with a quantity needs to be captured and represented in the quantity embeddings. However, such relationships cannot be effectively modeled using recurrent neural networks, which are commonly used in the existing math word solvers having deep learning methods. Graph2Tree introduces quantity cell graph and quantity comparison graph to capture and include these latent representations. The Quantity Cell Graph is built to associate informatively expressive words to quantity. The authors first extract associated verbs, nouns, adjectives, rates and units that describe a quantity in the problem statement. Next, a graph is constructed where the extracted expressive words are represented as neighbor nodes linked directly with the quantity. Finally, a neural network based model is used to learn the enhanced latent representation of the quantities based on the constructed Quantity Cell Graph.

The intuition behind introducing Quantity Comparison Graph is to retain the numerical qualities of the quantity. As we generally replace numerical quantities with their placeholders we tend to lose out on magnitude specific information which affects the order representation. The idea is to leverage certain heuristics to represent the relationships among quantities such that solution expression reflects a more realistic arithmetic order.

Methodology: Figure 2.10 shows the Graph2Tree framework. Graph2Tree first encodes the problem text input using bidirectional LSTM network and simultaneously constructs Quantity Cell Graph and Quantity Comparison Graph. The output of Bi-LSTM are seen as word-level representations and are

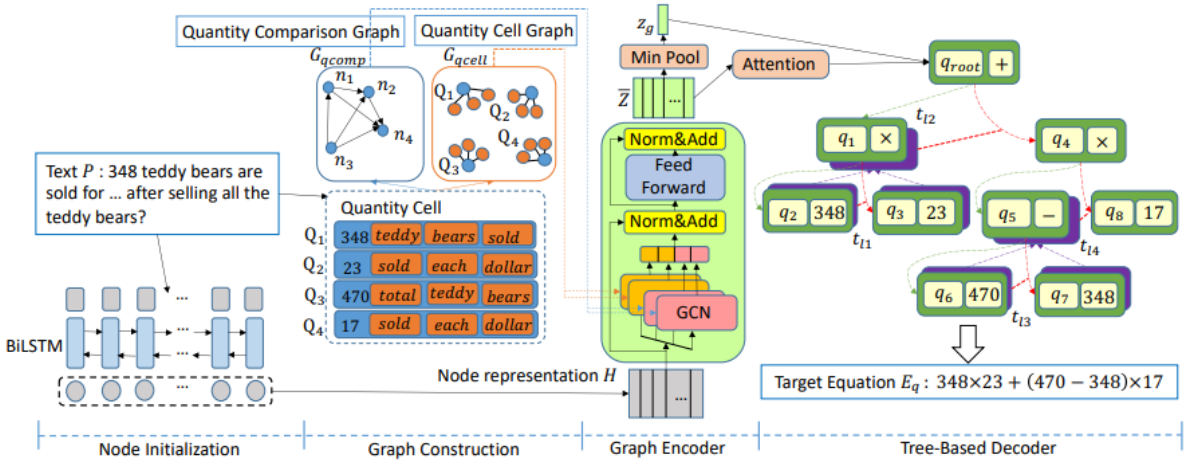

Figure 2.10: Overview of the Graph2Tree model architecture.

used for the representation of graph nodes. For both the constructed graphs, the node representations are together fed as input into a graph transformer to learn the graph representation of given math problem statement. The multi graph convolution network component of the graph transformer is slightly altered and modified to learn the graph representation based on the Quantity Cell Graph and Quantity Comparison graph. The final graph representation is enriched with the relationship information between the quantities and their numerical qualities. Pooling is then utilized to collect all nodes into a pool-based graph embedding vector as the output of the graph transformer. Finally, the output graph representation and the updated node representations are used as input to a tree-structure decoder to infer the final solution expression tree.

A quantity cell in a problem statement P consists of the following properties:

- **Quantity:** The numeric value associated with a quantity.
- **Associated Nouns:** The nouns related to the quantity in the dependency parse tree. Associated nouns are the nouns related by the number and preposition of relations.
- **Associated Adjectives:** Associated Adjectives are the adjectives related to quantity or associated nouns with the a modifier relation, which is detected by the dependency parser.
- **Associated Verbs:** For each quantity, the related verbs, associated Verbs, according to nsubj and dobj relations.
- **Units and Rates:** The nouns related to the associated nouns by preposition ”of” as the Unit. The nouns related with associated nouns which own the key words such as “each”, “every” and “per” are regarded as rates.

	MAWPS	Math23K	Math23K*
DNS	59.5	-	58.1
Math-EN	69.2	66.7	-
T-RNN	66.8	66.9	-
S-Aligned	-	-	65.8
GROUP-ATT	76.1	69.5	66.9
AST-Dec	-	69.0	-
GTS	82.6	75.6	74.3
IRE	-	76.7	-
Graph2Tree	**83.7**	**77.4**	**75.5**

Figure 2.11: Accuracy comparison between Graph2Tree and various baselines. Math23K* denotes 5-fold cross-validation.

If the quantity cell detection process is unable to identify any attributes, they use a window that is centered around the quantity to select neighboring words as the attributes.

Results: Table 2.11 compares the accuracy metric between Graph2Tree and other various baselines. The observations reflect that Graph2Tree outperforms all baselines in both MaWPS and Math23K. Table 2.9 through some examples illustrates the supremacy of Graph2Tree over GTS. This allows establishes the benefits of using a graph based encoder over standard Bi-LSTM based encoders.

2.2 Data Augmentation Methods

Data Augmentation refers to a set of techniques which are used to add additional training samples without creating new data [5]. These samples are created synthetically by modifying or altering existing samples. Data augmentation techniques were first explored in the field of computer vision where new image samples are augmented by adding noise and transforming them through primitive operations like rotation and changing colour channels [80]. These intuitive and straight forward operations are helpful as these are invariant to model outputs. However, data augmentation techniques are challenging to adapt for NLP tasks due to the discrete nature of textual data. In NLP, the input space is discrete and it is less obvious on how to modify samples to induce required diversity without modifying the target labels.

The purpose of performing data augmentation is to enhance the number of samples which can help in training and fine-tuning machine learning models on many downstream or low resource tasks where the number of training samples is very low and expensive to generate. These techniques are extremely helpful in the age of deep learning models where the number of model parameters are very high. We require large amounts of data to prevent over-fitting and developing the ability to handle diverse samples at inference time in the models.

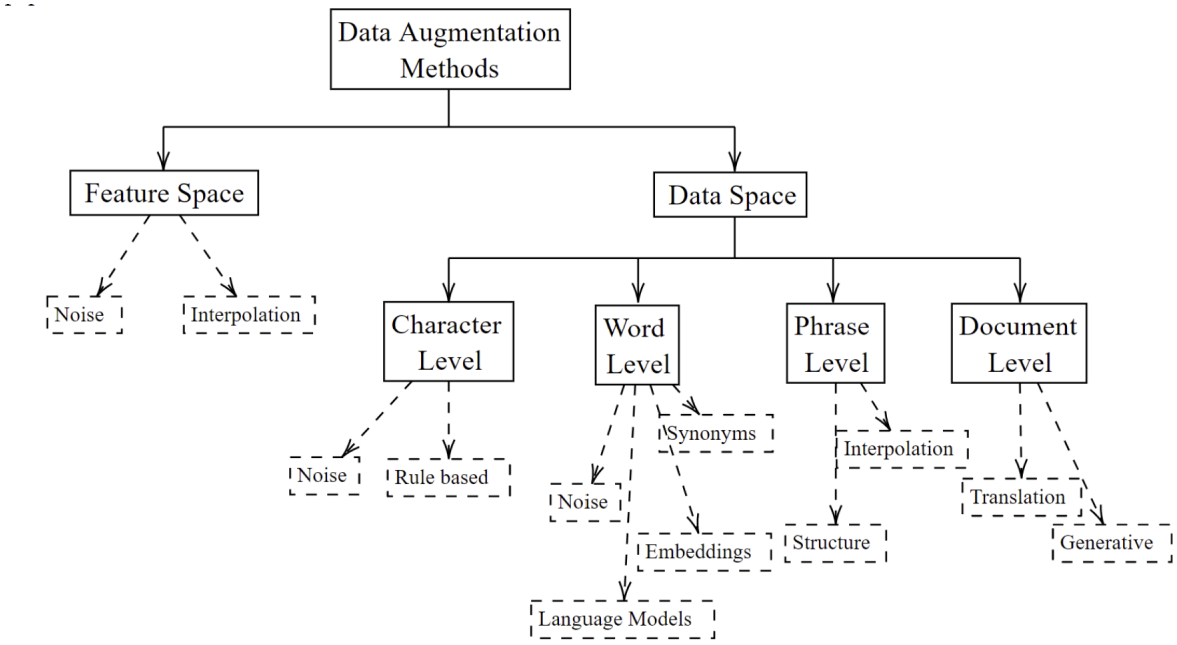

Figure 2.12: Categories of Data Augmentation methods for NLP Tasks.

Let's go through some of the popular text augmentation methods, particularly in the data space and also look at the filtration mechanism to filter out augmentations which do not meet the criteria.

2.2.1 Data Space

Augmentation in data space deals with transforming the input data in its original form that is, the readable textual form.

- **Character Based:** Character based augmentations are generally of two types: Noise based Inductions [6] and Rule based [13] inductions. In noise based inductions, artificial and natural noise is added to the words present in a sentence. Operations like random switching of letters, arbitrary deletion of letters, generating a permutation of the letters with initial and end letters fixed are utilized to augment the words. Natural noise covers misspelled words that commonly occur in natural language. Each word that is associated with a common error is replaced with the misspelled word and if there is more than one such errors, the mistake is then randomly sampled.
  Rule based inductions utilize regular expressions to create a template for changing the order of letters in words. These rules are not easy to create as they require deeper transformations to preserve the grammatical correctness, further these transformations are dependent on the language and its structure. The transformations of acronyms and short forms to their longer versions and vice - versa is a neat trick to achieve good augmentation results as shown in [13].

- **Word Based:** Word level transformations can further be categorized into the following types:

- **Synonym Replacement:** The use of synonym substitution is a common method for paraphrasing, which involves replacing certain words in text inputs with synonyms. One of the first applications of this technique in the field of data augmentation was introduced by [32], who substituted temporal expressions with potential synonyms from WordNet [47]. The authors argue that replacing a single token in a sentence typically preserves the semantics, and they propose replacing the headword (since temporal trigger words are often found there) in the context of a time expression recognition task. Synonym substitution can be a useful method for generating diverse but semantically similar variations of a text, which can be helpful for tasks such as data augmentation and evaluation.

- **Embedding Replacement:** Replacement methods for embedding aim to find words that fit as closely as possible into the textual context while also preserving the meaning of the text. To do this, the words in the instances are transformed into a latent representation space, where words with similar contexts are closer together. These latent spaces are based on the distributional hypothesis of distributional semantics, which is commonly implemented using embedding models. These models seek to capture the meaning of words based on their distribution or occurrence in a text corpus [22], [19], which is currently mostly implemented in the form of embedding models.

- **Noise Induction:** Noise induction methods can also be used in word replacement-based methods. For example, the method proposed in [79] includes two noise patterns: "uni-gram noising," in which words in the input data are replaced by another word with a certain probability, and "blank noising," in which words are replaced with an underscore. The authors found that using both of these patterns resulted in improved results in their experiments. Noise induction can be used to introduce variations in the input data, which can help to improve the robustness and generalization ability of the model.

- **Language model Replacement:** Language models represent language by predicting subsequent or missing words based on the previous or surrounding context (classical and masked language modeling, respectively). These models can be used, for example, to filter out unfitting words, as described in the work of [2]. The authors generate similar words using GloVe embeddings [53] and a counter-fitting method, and then use a language model to select only words with a high probability of fit. In contrast to embedding replacements that consider the global context, language models allow for more localized replacement [44]. Language models can also be used as the main augmentation method, as in the case of [31], who uses an LSTM language model to identify substitution words. However, it's worth noting that language models may not only substitute words with similar meaning, but also with words that fit the context in principle [31].

- **Phrase and Sentence Level:**

- **Structure Based:** Structure-based approaches for data augmentation may use certain features or components of a structure to generate altered texts. These structures can be based on grammatical formalities, such as dependency and constituent grammars or POS-tags. These approaches are therefore more limited to specific languages or tasks. For example, [85] focuses on augmenting datasets for POS-tagging in low-resource languages. They use a technique called "cropping" to shorten sentences by focusing on subjects and objects. Structure-based approaches can be useful for generating syntactically and semantically valid variations of a text, and they may be particularly useful in tasks where the structure of the language is important, such as in POS-tagging or parsing.
- **Interpolation Based:**

  In numerical analysis, interpolation is a method of constructing new data points from existing points [63]. While it is difficult to define interpolation in the data space of text, the substructure substitution (SUB²) method proposed by [62] can be considered as a form of interpolation in this context due to its similarity to interpolation methods in the feature space. SUB² substitutes substructures (such as dependents, constituents, or POS-tag sequences) of the training examples if they have the same labeled tag (for example, "a [DT] cake [NN]" in one instance can be replaced with "a [DT] dog [NN]" from another instance). The variant of SUB² adapted for classification views all text spans of an instance as structures and is constrained by replacement rules that can be combined or omitted altogether. This method can be used to generate diverse but semantically similar variations of a text, which can be helpful for tasks such as data augmentation and evaluation.

- **Document Level:**

  - **Translation Based:** Round-trip translation is a method of generating paraphrases using translation models. It involves translating a word, phrase, sentence, or document into another language (forward translation) and then translating it back into the source language (back translation) [1]. This approach is based on the idea that translations of texts can be variable due to the complexity of natural language [13], leading to multiple possibilities in terms of word choice and sentence structure. Round-trip translation has been found to be effective in preserving labels and producing high-quality paraphrases. When text is translated, the content is preserved while only stylistic features based on the author's style are excluded or changed [56]. This technique can be used to generate diverse variations of a text while preserving its meaning, which can be helpful for tasks such as data augmentation and evaluation.
  - **Generation Based:**

    Generative methods are becoming increasingly popular in recent data augmentation research. As the capabilities of language generation have significantly improved, current models are able to create very diverse texts and can therefore incorporate new information. [55]

introduce a variational autoencoder (VAE) based method for text generation in their system. VAEs are probabilistic neural network structures that consist of an encoder network, which transforms input data into a latent representation, and a decoder network, which transforms the latent representation back into the original data space. The authors differentiate between unconditional and conditional VAEs. With unconditional VAEs, separate text generation models are trained for each class, while with conditional VAEs, label information is fed into the model as an additional input. They also distinguish between sampling from the prior distribution, which leads to highly diverse instances, and sampling from the posterior distribution, which produces text that is more semantically similar to the training data. [42] also investigate VAEs for augmentation using the unconditional VAE and sampling from the prior distribution. VAEs can be a powerful tool for generating diverse and semantically rich variations of a text, which can be useful for tasks such as data augmentation and evaluation.

2.2.2 Filtering Mechanism

Filtering mechanisms are important for methods that are not perfectly label preserving. For example, [37] use a simple mechanism of removing generated instances based on the overlap of uni-gram words with the original equivalents. Other metrics, such as the Levenshtein distance, Jaccard similarity coefficient, or Hamming distance, could also be used for this purpose. [69] uses a similarity discriminator, originally proposed by [51], to measure the similarity of two sentences. The generative methods proposed by [12] filter instances using a classifier trained on the class data, which reduces the diversity of the samples. [5] improve this approach by using embeddings to measure the quality of the generated instances in relation to the class, and by incorporating human experts to determine the correct threshold. However, [81] propose a different filtering mechanism that does not require human assistance and is more sophisticated due to the inclusion of two perspectives. In general, [81] propose a generative method that is suitable for increasing the dataset size for question answering tasks. Filtering mechanisms can be useful for ensuring that generated instances are semantically similar to the original text and are suitable for the intended task.

2.3 Adversarial Attacks on Text

Adversarial samples are strategically generated examples that aim to alter a deep neural network's predictions. These samples are generated such that the modifications are imperceptible to the common user but will make the intelligent models modify their predictions or lead them to generate erroneous predictions. The process of constructing adversarial samples to fool an AI model is known as adversarial attacks. These type of attacks are generally used to test the robustness of an AI model. Adversarial samples are generally constructed by modifying the test samples. Broadly text adversarial attack can be categorized into:

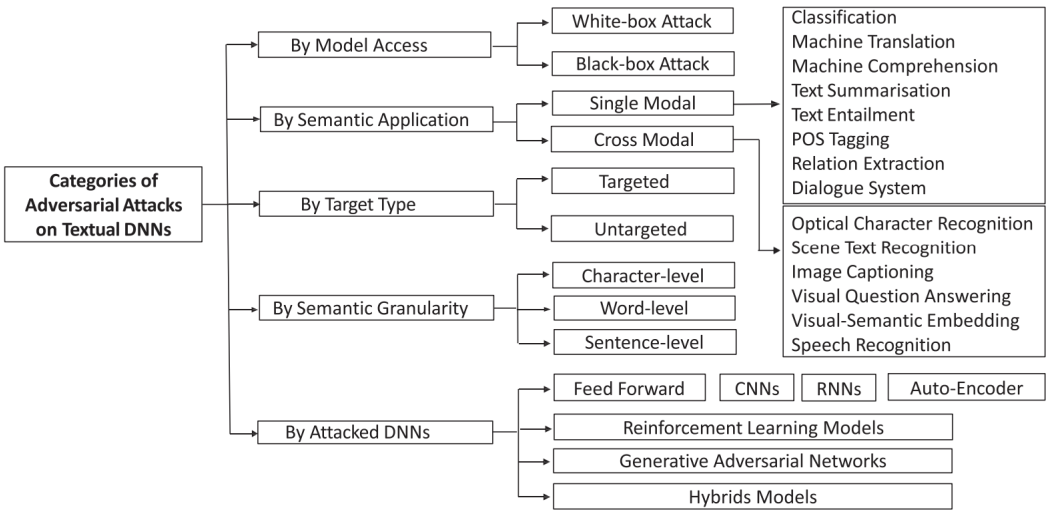

Figure 2.13: Categories of Adversarial attacks

- **White Box Attacks**: A white-box attack requires the access to the model's information, including its architecture, parameters, loss functions, activation functions, input and output data. White-box attacks typically approximate the worst-case attack for a particular model and input, incorporating a set of perturbations. This attack strategy is often very effective.
- **Black Box Attacks**: Black-box attack does not require the details of the neural networks, but can access the input and output. This type of attacks often rely on heuristics to generate adversarial examples, and it is more practical as in many real-world applications the details of the deep neural nets is a black box to the attacker.
- **Hard Label Attacks**: Hard label attacks are derived from black box attacks. These category of attacks do not have even the output layer distributions, but only know about the final predictions. These type of attacks are considered the toughest to execute and are extremely challenging to design.

2.3.1 Adversarial Attacks on common NLP Tasks

- **Natural Language Inference:** Natural Language Inference (NLI) [49] is a task that involves determining the relationship between two sentences (e.g., whether one sentence is entailed by the other). [21] sampled incorrectly classified examples and analyzed their potential sources of errors, which were then grouped into a typology of common error types. These error types served as the basis for constructing a stress test set to further evaluate whether NLI models can make real inferential decisions, or if they rely on sophisticated pattern matching. Previous research has found that current NLI models tend to identify the label by relying only on the hypothesis

[54], and similar results have been obtained by [59], who showed that a hypothesis-only model can outperform a set of strong baselines. [29] also asked humans to generate counterfactual NLI examples to better understand the causal features that encourage models to learn those features. NLI is an important task in natural language processing and understanding the potential sources of error and the features that influence performance can help improve the performance of models.

- **Question Answering:** [25] proposed generating adversarial question answering examples by concatenating an adversarial distracting sentence at the end of a paragraph. [48] constructed four new test sets for the Stanford Question Answering Dataset (SQuAD) and found that most question answering systems fail to generalize to this new data, suggesting the need for new evaluation metrics to better handle natural distribution shifts. Adversarial examples and shifts in the data distribution can be challenging for question answering systems and finding ways to better evaluate and address these issues can improve the performance and robustness of these systems.

- **Machine Translation:** [6] found that character-based neural machine translation (NMT) models are brittle under noisy data, such as typos or misspellings. Data augmentation techniques that introduce artificially generated grammatical errors [4] or synthetic noises [67] [28] can make these systems more robust to these types of patterns. However, [70] proposed a different approach by limiting the input space of characters so that the models are more likely to correctly handle data typos and erroneous spellings. These approaches aim to improve the robustness of NMT models to noisy data, which can be important for practical applications where the input may not always be clean and error-free.

- **Natural language Generation:** Existing research has found that text generation models can be vulnerable to robustness issues, such as positional bias [27] and layout bias [34] in text summarization models, and a lack of faithfulness and factuality [34]. Data-to-text models can also generate texts that are not supported by the data [51]. [82] pointed out the limitations of current automatic evaluation metrics and proposed new metrics to better align the quality of generation with human judgments. These issues highlight the importance of addressing robustness in text generation models, as well as the need for more effective evaluation methods to accurately assess the quality of generated text.

Chapter 3

Adversarial Examples for Evaluating Math Word Problem Solvers

3.1 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 3.1. 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 [33, 46] to complex problems that require solving a system of equations [24, 60]. In this chapter, 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 [23, 36] and deep learning based solvers . However, recent deep learning based approaches [72, 78, 82] have established their superiority over statistical learning based solvers. Here, we will briefly review some recent MWP solvers. Initially, [72] modelled the task of MWP as a sequence to sequence task and utilized Recurrent Neural Nets (RNNs) to learn problem representations. Building upon this, [10] focused on learning representations for mathematical operators and numbers, [78, 71] utilized tree structure to develop decoders for MWP solvers. More recently, to learn accurate relationship between numerical quantities and their attributes [82] modelled encoder as a graph structure.

All such MWP solvers have achieved high performance 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, [52] proposed a challenge test set called SVAMP which demonstrate that existing MWP solvers rely on shallow heuristics to achieve high performance. Instead

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

of relying on standard accuracy metrics, many works have used adversarial examples [64, 50] 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 [25] authors crafted adversarial attacks to test the robustness of QA systems. Prior works in [20, 45] uses adversarial examples to show deficiencies of NLI models. Similarly, [15, 9] uses adversarial examples to develop robust dialogue and neural machine translation models. Recently, there has been a plethora of work [16, 2, 26, 41, 40] 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 chapter, 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 [2] are not suitable for MWP as the fluency of the problem statement is not preserved. Similarly, paraphrasing techniques like back-translation [43] 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.

3.2 Proposed Approach

3.2.1 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 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.

3.2.2 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 $\{S_1, S_2..S_k\}$.

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

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 3.1. 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.

3.2.3 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 [23] 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. Therefore, it is used to evaluate whether existing MWP solvers rely on specific keywords or patterns to generate equations. Figure 3.1 shows all the steps followed by the proposed algorithm above to generate paraphrases.

[2]https://spacy.io/

Algorithm 1 Sentence Paraphrasing

Input: Problem text $\mathcal{P}$, $\mathcal{M}$ is Paraphrase model
Output: Adversarial text $\mathcal{P}^*$

$\mathcal{P}^* \leftarrow \mathcal{P}$
$y_{orig} \leftarrow \mathbf{F}(\mathcal{P})$
for S_i *in* $\mathcal{P}$ **do**
    $C \leftarrow \mathcal{M}(S_i)$
    for c_j *in* C **do**
        if $h_{i,orig} == h_{i,j}$ **then**
            $paraphrases.add(c_j)$
    $paraphrases.add(S_i)$
    $candidates.add(paraphrases)$
for c_k in $Combinations(candidates)$ **do**
    $y_{adv} \leftarrow \mathbf{F}(c_k)$
    if $y_{adv} \neq y_{orig}$ **then**
        $\mathcal{P}^* \leftarrow c_k$
        end

Figure 3.1: Algorithm showcasing the set of steps to perform sentence paraphrasing.

3.3 Experiments

3.3.1 Datasets and Models

We evaluate the robustness of three state-of-the-art MWP solvers: (1) *Seq2Seq* [72] having an LSTM encoder and an attention based decoder. (2) *GTS* [78] having an LSTM encoder and a tree based decoder and (3) *Graph2tree* [82] 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 [72] or they have problems of higher complexities [24] . We conduct experiments across the two largest available English language datasets satisfying our requirements: (1) *MaWPS* [33] containing $2,373$ problems (2) *ASDIV-A* [46] containing $1,213$ problems. Both datasets have MWPs with linear equation in one variable.

3.3.2 Experimental Setup

We trained the three MWP solvers from scratch as implemented in baseline paper [72] on the above two datasets using 5-fold cross-validation as followed in [82]. The original accuracies obtained on the datasets are shown in Table 2. We used [83] 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 3.4.3) to verify the quality of generated adversarial examples.

Dataset	Evaluation Type	Seq2Seq	GTS	Graph2Tree
MaWPS	Original	53.0	82.6	83.7
	Question Reordering	**18.2**	**32.3**	**35.6**
	Sentence Paraphrasing	**10.5**	**22.7**	**25.5**
ASDIV-A	Original	54.5	71.4	77.4
	Question Reordering	**17.5**	**30.5**	**33.5**
	Sentence Paraphrasing	**13.2**	**21.2**	**23.8**

Table 3.2: Results of MWP Solvers on adversarial examples.

3.3.3 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. Hyper-parameter values were not modified, and we follow the recommendations of the respective models. We chose the number of candidate paraphrases m used in Algorithm in Figure 3.1 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.

3.3.4 Results

Table 3.2 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.

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

Table 3.3: Accuracy of MWP solvers with adversarial training on our proposed methods. Adv and BERT represent models trained from scratch and BERT embeddings respectively.

3.4 Analysis

3.4.1 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 3.3. 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.

3.4.2 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.

3.4.3 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 are shown in Table 3.4.

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 3.4: Human Evaluation scores on MaWPS and ASDiv-A datasets.

3.5 Conclusion

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.

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.

Chapter 4

Data Augmentation for Math Word Problem Solvers

4.1 Introduction

In recent years, the challenge of solving MWP has gained much attention in the NLP community as it needs the development of commonsense multi step reasoning with numerical quantities. With the rise of deep learning, performance of math solvers has also increased significantly over the years [72, 82]. However, recent analysis conducted in [52] show that these deep learning based solvers rely on shallow heuristics to solve vast majority of problems. They curated adversarial examples and SVAMP challenge set respectively to infer that MWP solvers (1) do not understand the relationship between numbers and their associated entities, (2) do not focus on the question text and (3) ignore word order information. In this chapter, we first conduct experiments to establish that the above drawbacks are due to the limited size and diversity of problems present in the existing MWP datasets. Next, we propose various augmentation methods to create diverse and large number of training examples to mitigate these shortcomings. Our methods are focused on: (1) Increasing the number of problems in the existing datasets and (2) enhancing the diversity of the problem set.

Training deep neural models effectively requires large number of data points [38]. Constructing large datasets which are annotated, labeled and have MWPs of similar difficulty level is a very expensive and tedious task. To address these key challenges, we resort to data augmentation techniques. Our motivation behind generating augmentations is that humans require sufficient practice to understand MWPs. Humans learn to solve MWPs by going through a variety of similar examples and slowly become capable enough to tackle variations of similar difficulty levels. We aim to generate augmentations such that sufficient linguistic variations of a similar problem are present in the dataset. These variations will make the solver more robust in tackling MWP, increase their reasoning ability and numerical understanding.

Data augmentation for MWPs is a challenging task as we need to preserve the equation labels while generating new samples. The generated samples should be (1) semantically similar to their original counterpart, (2) must have the same numerical values and preserve relationship with their respective entities and (3) should maintain the same sequence of events in the problem text. Existing augmentation

Original Problem
Problem: Nancy grew 8 potatoes. Sandy grew 5 potatoes.How many potatoes did they grow in total ?
True Equation: X = 8+5
Paraphrasing Method
Problem: How many potatoes did they grow in all given that nancy grew 8 potatoes and sandy grew 5 potatoes.
Equation Label: X = 8+5
Substitution Method
Problem: Dwight grew 8 potatoes. Juliette grew 5 potatoes.
How many potatoes did they grow together ?
Equation Label: X = 8+5

Table 4.1: A MWP and its augmentation examples generated by our methods with preserved equation labels. Blue and Violet colours denote the changes made after the primary stage and secondary stage respectively.

methods [73] cannot be directly applied due to the above mentioned reasons. Our methods can be broadly classified as follows:

- **Paraphrasing Methods:** It generates variations of the question text by re-statement such that the semantic and syntactic meaning along with the equation labels is preserved.

- **Substitution Methods:** These methods generate variations of the problem statement by identifying and substituting some of the keywords such that the augmentations are semantically and syntactically correct.

To ensure high quality augmentations, we propose a selection algorithm which selects samples that have high similarity with original problem and incur high loss values when tested on existing solvers. This algorithm helps selecting only those samples that can make existing solvers more robust. Further, we also verify the validity and the quality of generated augmentations through human evaluation.

Most of the existing MWP datasets are either in languages other than English or contain problems of varying difficulty levels [33, 72, 24, 3, 46]. We focus on strengthening existing English language datasets which can facilitate the development of better MWP solvers. We consider datasets containing MWP that can be solved using linear equations in one variable. These datasets include MaWPS [33] and ASDiv-A [46] both having $2,373$ and $1,213$ problems respectively. Following are the key contributions made in this chapter:

- To the best of our knowledge, this is the first work that extensively evaluates data augmentation techniques for MWP solving. This is the first attempt to generate MWP problems automatically without manual intervention.
- Accuracy of the state of the art solvers increases after training on the proposed augmented dataset. This demonstrates the effectiveness of our methods. To verify the validity of generated augmentations we conduct human evaluation studies.
- We increase the diversity of the training dataset through augmentations and obtain comparatively better results than state-of-the-art solvers on the SVAMP challenge set.

4.2 Related Work

Text Data Augmentation: To effectively train deep learning models, large datasets are required. Data augmentation is a machine learning technique that artificially enlarges the amount of training data by means of label preserving transformations [66]. [38] hypothesize that textual data augmentation would only be helpful if the generated data contains new linguistic patterns that are relevant to the task and have not been seen in pre-training. In NLP, many techniques have been used for generating augmentations, [73] introduced noise injection, deletion, insertion and swapping of words in text. [59] used recurrent neural networks and generative adversarial networks for short-text augmentation. Other frequently used methods include inducing spelling mistakes [6], synonym replacement [84], identifying close embeddings from a defined search space [2], round trip translations [61], paraphrasing techniques [35] and words predicted by language model [31] among many others. These methods are specific to the task at hand and needs to be adapted such that the generated augmentations bring diversity in the concerned dataset.

4.3 Proposed Augmentation Approach

Data augmentation generates new data by modifying existing data points through transformations based on prior knowledge about the problem domain. We introduce carefully selected transformations on well known text augmentation techniques to develop examples suited for the task of MWP. These transformations help in increasing the diversity and size of problem set in existing datasets.

4.3.1 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 $\mathcal{E}_\mathcal{P}$ from $\mathcal{P}$ such that the equation consists of numbers from $\mathcal{P}$, desired numerical constants and mathematical operators from the set $\{/, *, +, -, =, (,)\}$. Let $\mathcal{F} : \mathcal{P} \rightarrow \mathcal{E}_\mathcal{P}$ be an MWP solver where $\mathcal{E}_\mathcal{P}$ is

Dataset	Eval Type	Seq2Seq	GTS	Graph2Tree
MaWPS	True	84.6	87.5	88.7
	Word Deletion	**80.2**	**81.5**	**77.3**
	Question Reordering	**77.4**	**82.0**	**80.2**
	Sentence Shuffling	**77.0**	**60.4**	**66.4**
	Word Reordering	**54.9**	**34.8**	**39.3**
ASDiv-A	True	70.6	80.3	82.7
	Word Deletion	**60.2**	**61.3**	**56.7**
	Question Reordering	**58.7**	**52.4**	**54.1**
	Sentence Shuffling	**56.2**	**59.3**	**60.7**
	Word Reordering	**47.1**	**32.3**	**34.6**

Table 4.2: Performance of solvers on modified test sets. True represents unaugmented test set.

the equation to problem $\mathcal{P}$. Our task is to generate augmented problem statement $\mathcal{P}^*$ from the original input $\mathcal{P}$ such that $\mathcal{P}^*$ is: (1) semantically similar to the initial input $\mathcal{P}$, (2) preserves the sequence of events in the problem statement, (3) keeps the numerical values intact and (4) the solution equation is same as $\mathcal{E}_{\mathcal{P}}$.

4.3.2 Deficiencies in Existing Models

As showcased by [52], existing MWP solvers trained on benchmark datasets like MaWPS and ASDiv-A focus their attention only on certain keywords in the problem statement and do not pay much heed to the question text. We further show that even after performing significant transformations on the test set such as (1) dropping the question text, (2) randomly shuffling the sequence of sentences, (3) random word deletion, and (4) random word reordering, the solvers are still able to produce correct equations. Upon introducing these transformations we should expect a very high drop in accuracy values as the transformed problems are now distorted. Surprisingly, the decrease in accuracy scores is relatively very less than expected as shown in Table 4.2. We only observe a relatively moderate drop for word reordering. From this analysis, we can say that instead of focusing on the sequence of events, question text and semantic representation of the problem, solvers pick word patterns and keywords from the problem statement. We hypothesize that the drop in accuracy for word reordering experiment indicates that the solvers try to identify a contiguous window of words having some keywords and numbers in them, and

generates equation based on these keywords. We further probe on this hypothesis by visualizing the attention weights in the experiment section.

4.3.3 Augmentation Methods

A MWP can also be expressed as $\mathcal{P} = (S_1, S_2..S_k, Q)$ where Q is the question and $(S_1, S_2..S_k)$ are the sentences constituting the problem description. To mitigate the deficiencies in MWP solvers, we propose a two stage augmentation paradigm consisting of primary and secondary stage. In primary stage, we generate base augmentation candidates which then proceed to the secondary stage and get modified accordingly to become potential candidates. After identifying the potential candidates, we filter out the best candidates using proposed candidate selection algorithm. Table 4.1 shows changes in MWP after primary and secondary stage. Following are the details:

Primary Stage: In the primary stage, our focus is on inducing variations in the question text Q of a given problem statement $\mathcal{P}$. For this, we first generate n base candidates $\{b_1, b_2, ..., b_n\}$ from Q using $T5$ paraphrasing model [57]. The key intuition behind this step is to ensure that each augmentation of a given problem has a different question text. This will empower the solver to learn variations from the question text as well.

Secondary Stage: After the generation of base candidates, we implement augmentation methods to generate potential candidates. These methods although well known, require careful tuning to adapt for MWP generation. Table 4.3 showcases MWP examples and their generated augmentations. Detailed description of these techniques follow.

4.3.3.1 Paraphrasing Methods

Paraphrasing has proved to be an effective way of generating text augmentations [74]. It generates samples having diverse sentence structures and word choices while preserving the semantic meaning of the text. These additional samples guide the model to pay attention to not only the keywords but its surroundings as well. This is particularly beneficial for the task of MWP solving, where most of the problem statements follow a general structure.

Problem Reordering: Given original problem statement $\mathcal{P} = (S_1, S_2, ...S_k, Q)$, we alter the order of problem statement such that $\mathcal{P}^* = (Q, S_1, S_2, ..., S_k)$. To preserve the semantic and syntactic meaning of problem statement we use filler phrases like 'Given that' and 'If-then'. To make these paraphrases more fluent, we use named entity recognition and co-reference resolution to replace the occurrences of pronouns with their corresponding references. Please note that this method is better than random shuffling of sentences as it preserves the sequence of events in the problem statement.

Round Trip Translations: Round trip translations, more commonly referred as back-translation [76] is an interesting method to generate paraphrases. This idea has evolved as a result of the success of machine translation models [75]. In this technique, sentences are translated from their original language to foreign languages and then translated back to the original language. This round trip can be between

Category	Augmentation Method	Example
Paraphrasing Methods	**Round trip Translation**	**Original:** The schools debate team had 4 boys and 6 girls on it. If they were split into groups of 2, how many groups could they make ? **Augmented:** The school discussion group consisted of 4 boys and 6 girls. If they are divided into groups of 2 . How many groups could they have created ?
	Problem Reordering	**Original:** Lucy has an aquarium with 5 fish . She wants to buy 1 more fish . How many fish would Lucy have then ? **Augmented:** If lucy has an aquarium with 5 fish and she wants to buy 1 more fish then how many fish would lucy have ?
Substitution Methods	**Fill Masking**	**Original:** There are 8 walnut trees currently in the park . Park workers will plant 3 more walnut trees today . How many walnut trees will the park have when the workers are finished ? **Augmented:** There are 8 walnut trees currently in the park . Park workers will plant 3 more walnut trees soon . How many walnut trees will the park have after the workers are finished ?
	Named-Entity Replacement	**Original:** Sally found 7 seashells , Tom found 12 seashells , and Jessica found 5 seashells on the beach . How many seashells did they find together ? **Augmented:** Edd found 7 seashells , Alan found 12 seashells , and Royal found 5 seashells on the beach . How many seashells were found together ?
	Synonym Replacement	**Original:** Katie 's team won their dodgeball game and scored 25 points total . If Katie scored 13 of the points and everyone else scored 4 points each , how many players were on her team ? **Augmented:** Katie's group won their rumble game and scored 25 points total . If Katie scored 13 of the points and all else scored 4 points each, How many players was on her group ?

Table 4.3: Augmentation examples from all proposed methods. Coloured text represents the changes in problem statement.

multiple languages as well. The motivation behind using this technique is to utilize the different structural constructs and linguistic variations present in other languages.
Back-translation is known to diverge uncontrollably [65] for multiple round trips. This may lead to change in the semantics of the problem statement. Numerical quantities are fragile to translations and their order and representation may change. To overcome these challenges, we worked with languages that have structural constructs similar with English. For instance, languages like Finnish which are gender neutral, can become problematic as they can lead to semantic variance in augmented examples. To preserve numerical quantities, we replace them with special symbols and keep a map to restore numerical quantities in the generated paraphrases. We have used the following round trips:

English - Russian - English: Although Russian is linguistically different from English, we still chose it as word order does not affect the syntactic structure of a sentence in Russian language [68]. For single round trip, we preferred Russian as it has the potential to generate different paraphrase structures.

English - German - French - English: German and French are structurally similar to English language [30], we chose them for multiple round trips to both maintain semantic in-variance and induce minor alterations in the paraphrases.

4.3.3.2 Substitution Methods

In this class of methods, the focus is on generating variations of the problem statement by identifying and substituting some of the keywords such that the augmentations are semantically and syntactically correct, with the equation labels preserved. Substitution is effective for MWP solving as it guides the solvers focus away from certain keywords, allowing it to distribute its attention and generalize better. We propose the following methods:

Fill-Masking: In this technique, we model the challenge of generating candidates as a masked language modelling problem. Instead of randomly choosing words for masking, we use part of speech tags to focus on nouns and adjectives, preferably in the vicinity of numerical quantities. We replace these identified keywords with mask tokens. These masked candidate sentences are then passed through a masked language model [14] and suitable words are filled in masked positions to generate our candidate sentences.

Synonym Replacement: In this method, after stop-word removal, we select keywords randomly for substitution. Unlike fill-mask technique, where masked language models were deployed, here we use Glove embeddings [53] to find the top k candidates that are close synonyms of the keywords. To ensure syntactic correctness in candidate sentences, we maintain the part of speech tags for the substitute candidates. These synonyms are then used to substitute the keywords in the problem statement and generate augmented candidates.

Named-Entity Replacement: A common occurrence in MWP is the usage of named entities. These entities play a crucial role in stating the problem statement, but the solution equations do not change

on altering these entities. Following this insight, we first identify the named entities[1] such as person, place and organizations present in the problem statement. Then we replace these named entities with their corresponding substitutes, like a person's name is replaced by another person's name to generate the potential candidates.

Table 4.4 reports the statistics of augmented datasets on both MaWPS and ASDiv-A. All the techniques described in paraphrasing and substitution methods are used for generating the potential candidates for a problem statement. After generation of the potential candidates for augmenting a problem statement, the best possible candidate is selected by using the algorithm depicted in figure 4.1. Key motivation behind developing this algorithm is to select candidates on which the solver does not perform well and which are similar to the original problem statement.

We use negative log likelihood as the loss function $\mathcal{L}$ and Sentence-BERT [58] fine tuned on MWP equation generation task as sentence embedding generator $\mathcal{S}$. We calculate the similarity of each candidate embedding with the original problem representation using cosine similarity as shown in Line 3 of the algorithm. Further, for each candidate sentence, we evaluate their loss values and select the candidate with the maximum mean normalized loss and similarity score.

Dataset	**Problem Size**	**Vocabulary Size**
MaWPS	2,373	2,632
ASDiv-A	1,213	2,893
Paraphrase	5,909	3,832
Substitution	6,647	3,923
Combined-MaWPS	**10,634**	**5,626**
Combined-ASDiv	**5,312**	**6,109**

Table 4.4: Statistics of augmented dataset compared with MaWPS and ASDiv-A. Combined-Dataset represents combination of Paraphrase and Substitution methods.

4.4 Experiments

Datasets and Models: To showcase the effectiveness of proposed augmentation methods, we select three state-of-the-art MWP solvers: (1) *Seq2Seq* [72] having an LSTM encoder and an attention based

[1]https://www.nltk.org/

Algorithm 1 MWP Candidate Selection Algorithm

Requires: $\mathcal{M}$ is augmentation method, $\mathcal{S}$ is similarity model, $\mathcal{F}$ is solver model, $\mathcal{L}$ is Loss function.
Input: Problem text $\mathcal{P}$
Output: Augmented Text $\mathcal{P}^*$

$\mathcal{E}_P \leftarrow \mathcal{F}(\mathcal{P})$
$Candidates \leftarrow \mathcal{M}(\mathcal{P})$
for C_j *in* $Candidates$: **do**
    $S_j \leftarrow \mathcal{S}(C_j, \mathcal{P})$
    $L_j \leftarrow (\mathcal{L}(C_j) - \mathcal{L}(P))/\mathcal{L}(P)$
    $CandidateScore.add(S_j * L_j)$
$\mathcal{P}^* = \arg\max_{C_j} CandidateScore(C_j)$
end

Figure 4.1: Algorithmic steps showcasing the procedure for performing candidate selection algorithm.

decoder. (2) *GTS* [78] having an LSTM encoder and a tree based decoder and (3) *Graph2tree* [82] consists of a both tree based encoder and decoder. *Seq2Seq* serves as our base model for experimentation. Many existing datasets are not suitable for our analysis as either they are in Chinese [72] or they have problems of higher complexities [24] . We conduct experiments across the two largest available English language datasets satisfying our requirements: (1) *MaWPS* [33] containing $2,373$ problems (2) *ASDiv-A* [46] containing $1,213$ problems. Both datasets have MWPs with linear equation in one variable.

Experiment Setup: We train and evaluate the three solvers on both MaWPS and ASDiv-A using five fold cross validation. Evaluation is conducted on both original and augmented datasets. We use the same hyperparameter values as recommended in the original implementation of these solvers. Further, each solver has been trained from scratch and by using BERT embeddings [14]. We also evaluate the models on *SVAMP* [52] challenge set. This test set has been designed specifically to examine the robustness and adaptability of the solvers. Ablation studies have been conducted to assess the effectiveness of candidate selection algorithm and augmentation techniques.

4.4.1 Results and Analysis

Table 4.5 shows the result of proposed methods. These results have been reported on BERT embeddings. Table 4.11 shows a comparison between training from scratch and using BERT embeddings. By training these state-of-the-art models on the augmented dataset we achieve better results for both MaWPS and ASDiv-A. On average, we were able to increase the accuracy significantly by more than

Dataset	Eval Type	Seq2Seq	GTS	Graph2Tree
MaWPS	True	84.6	87.5	88.7
	Paraphrasing	**88.3**	**90.4**	**92.6**
	Substitution	**89.2**	**89.7**	**91.7**
	Combined	**91.3**	**92.6**	**93.5**
ASDiv-A	True	70.6	80.3	82.7
	Paraphrasing	**75.6**	**84.2**	**83.6**
	Substitution	**73.2**	**83.3**	**84.1**
	Combined	**78.2**	**85.9**	**86.3**

Table 4.5: Results of augmentation methods. True is unaugmented dataset, Combined is combination of Paraphrasing and Substitution methods.

Problem 1: Ricardo was making baggies of cookies with 5 cookies in each bag. If he had 7 chocolate chip cookies and 3 oatmeal cookies, how many baggies could he make ?

Solution Equation: X = (7+3)/5

Pre Augmentation Equation: X = (7/3)/3

Post Augmentation Equation: X = (7+3)/5

Problem 2: For halloween Destiny bought 9 pieces of candy. She ate 3 pieces the first night and then her sister gave her 2 more pieces. How many pieces of candy does Destiny have now ?

Solution Equation: X = 9-3+2

Pre Augmentation Equation: X = ((9+3-3

Post Augmentation Equation: X = (9+3-2)

Problem 3 : Audrey needs 6 cartons of berries to make a berry cobbler. She already has 2 cartons of strawberries and 3 cartons of blueberries. How many more cartons of berries should Audrey buy ?

Solution Equation: X = 6-2-3

Pre Augmentation Equation: X = (6-(2)+3)

Post Augmentation Equation: X = 6-(2+3)

Table 4.6: Examples illustrating equation results before and after training on the full augmented dataset.

five percentage points. Both paraphrasing and substitution methods have performed well independently and in combination. Further, we conduct ablation studies to analyze the performance of each augmentation method. In Table 4.6 we illustrate some examples on which existing models generate incorrect equations. However, after being trained with augmented dataset they generate correct equations. Additionally, in Problem 2 the base model generates syntactically incorrect solution, but post augmentation it generates syntactically correct equation. These examples show the increased robustness and solving abilities of solvers.

Attention Visualizations: Through this investigation, we aim to ascertain our hypothesis that to generate equations MWP solvers focus only on certain keywords and patterns in a region. They ignore essential information like semantics, sequence of events and content of the question text present in the problem statement. In Table 4.7, we show some sample problem statements with their attention weights. These weights are generated during the decoding process using Luong attention mechanism [39]. Moreover, to illustrate the effectiveness of our augmentation techniques, we show the distribution of attention weights for models trained on the augmented dataset. We can infer from the examples showcased in Table 4.7 that before augmentation the focus of the solver is limited to a fixed region around numerical quantities and it does not pay heed to the question text. However, after training on the augmented dataset the solver has a better distribution of attention weights, the weights are not localised and and the model is also able to pay attention on the question text.

Ablation Studies: To assert the effectiveness of our methods, we conduct the following ablations:

- ***Candidate Selection Algorithm:*** For testing the usefulness of candidate selection algorithm, we compare it with a random selection algorithm. In this, we randomly select one of the possible candidates as augmented problem statement. We evaluate the accuracy of models trained on the augmented datasets, generated using both the algorithms.Result in Table 4.8, shows that candidate selection algorithm performs better than random selection algorithm and this demonstrates the effectiveness of our algorithm.

- ***Augmentation Methods:*** To examine the effectiveness of proposed augmentation techniques, we evaluate the models on each of the proposed techniques independently and report the results in Table 4.9. Although, all the methods contribute towards increase in accuracy but Round trip translations and synonym replacement perform marginally better than others. This behaviour can be linked to the structural diversity and keyword sensitivity that round trip translations and synonym replacement bring respectively [18].

SVAMP Challenge Set: SVAMP [52] is a manually curated challenge test set consisting of $1,000$ math word problems. These problems have been cherry picked from MaWPS and ASDiv-A, then altered manually to modify the semantics of question text and generate additional equation templates. This challenge set is suitable for evaluating a solver's performance as it modifies problem statements such

Original Problem: A magician was selling magic card decks for 2 dollars each . If he started with 25 decks and by the end of the day he had 4 left, how much money did he earn ? Mean attention values: 0.34 0.11 0.09 Augmented Problem: A magician was selling magic card decks for 2 dollars each. If he started with 25 decks and by the end of the day he had 4 left, how much money did he earn ? Augmented mean attention values : 0.19 0.18 0.15
Original Problem: There are 18 pencils in the drawer and 6 pencils on the desk. Dan placed 4 pencils on the desk. How many pencils are now there in total ? Mean attention values: 0.21 0.16 0.06 Augmented Problem: There are 18 pencils in the drawer and 6 pencils on the desk. Dan placed 4 pencils on the desk. How many pencils are now there in total ? Augmented mean attention values : 0.29 0.19 0.12
Original Problem: Dan has 12 violet marbles, he gave Mary 4 of the marbles .How many violet marbles does he now have ? Mean attention values: 0.23 0.21 0.17 Augmented Problem: Dan has 12 violet marbles , he gave Mary 4 of the marbles. How many violet marbles does he now have ? Augmented mean attention values : 0.23 0.18 0.11
Original Problem: Angela has 7 tickets. Annie gives Angela 5 more. How many tickets does Angela have in all ? Mean attention values: 0.30 0.19 0.15 Augmented Problem: Angela has 7 tickets. Annie gives Angela 5 more. How many tickets does Angela have in all ? Augmented mean attention values : 0.29 0.21 0.14
Original Problem: Maria had 5 bottles of water in her fridge . If she drank 1 of them and then bought 2 more, how many bottles would she have ? Mean attention values: 0.48 0.14 0.04 Augmented Problem Maria had 5 bottles of water in her fridge . If she drank 1 of them and then bought 2 more, how many bottles would she have ? Augmented mean attention values : 0.23 0.17 0.11

Table 4.7: Examples illustrating distribution of top three attention weights before and after training on the full augmented dataset.

Method	Evaluation Type	Seq2Seq	GTS	Graph2Tree
Random Selection Algorithm	True	84.6	87.5	88.7
	Paraphrasing	**85.3**	**88.1**	**89.2**
	Substitution	**86.8**	**87.3**	**87.9**
	Combined	**87.0**	**89.2**	**89.5**
Candidate Selection Algorithm	True	84.6	87.5	88.7
	Paraphrasing	**88.3**	**90.4**	**92.6**
	Substitution	**89.2**	**89.7**	**91.7**
	Combined	**91.3**	**92.6**	**93.5**

Table 4.8: Ablation Study for Random Selection Algorithm and Candidate Selection Algorithm.

that solver's generalization can be checked. The results are shown in Table 4.10. Although, our proposed augmented dataset has very limited equation templates, still it performs comparatively better than state-of-the-art models on SVAMP challenge set. This result signifies the need for a larger and diverse dataset with enhanced variety of problems. Further, it demonstrates the effectiveness of our method which is able to perform better on SVAMP test set and increase model's accuracy despite the challenges.

BERT Embeddings: We train the solvers in two different settings, using pre-trained BERT base embeddings and training from scratch. We chose BERT specifically as we require contextual embeddings which could be easily adapted for the task of MWP. Moreover, existing models have also shown results using BERT and it would be fair to compare their performances when trained using similar embeddings. Results obtained are shown in Table 4.11. We observe that for solver's trained using BERT, accuracy is higher than models trained from scratch.

Human Evaluation: To verify the quality of augmented examples, we conduct human evaluation. The focus of this evaluation is: (1) To check if the augmentations will result in the same linear equation as present in the original problem statement, (2) To evaluate if the numerical values for each augmentation example is preserved, (3) Evaluate each sample in the range 0 to 1 for its semantic similarity with the original problem statement, (4) On a scale of 1 to 5 rate each augmented example for its grammatical correctness. We conduct the human evaluations on randomly shuffled subsets consisting of around 40% of the total augmented examples for both the datasets. This process is repeated three times with different subsets, five human evaluators evaluate each example in all subsets, and the mean results are computed as shown in Table 4.12.

Augmentation	Seq2Seq	GTS	Graph2Tree
True	84.6	87.5	88.7
Round trip Translations	**86.5**	89.1	**91.6**
Problem Reordering	85.9	88.4	90.7
Fill Masking	84.8	87.2	89.1
Synonym Replacement	85.2	**90.1**	91.2
Named Entity Replacement	86.1	88.3	89.7

Table 4.9: Result of Ablation study for each augmentation method. True represents unaugmented MaWPS dataset.

Augmentation	Seq2Seq	GTS	Graph2Tree
True	37.5	39.6	41.2
MaWPS(P+S)	39.2	40.1	42.3
ASDiv-A(P+S)	37.8	40.4	42.1
Combined	**40.2**	**41.3**	**43.8**

Table 4.10: Result of augmentations on SVAMP Challenge Set. P and S represent paraphrasing and substitution methods. Combined represents augmented MaWPS and ASDiv-A. True is combined MaWPS and ASDiv-A.

4.5 Future Work and Conclusion

We showcase that the existing MWP solvers are not robust and do not generalize well on even simple variations of the problem statement. In this paper, we first conduct experiments to showcase that this behaviour is mainly associated with the limited size and diversity present in existing MWP datasets. Next, we propose several data augmentation techniques broadly categorized into *Substitution* and *Paraphrasing* based methods. By deploying these methods we increase the size of existing datasets by five folds. Extensive experiments on two benchmark datasets across three state-of-the-art MWP solvers shows that proposed methods increase the generalization and robustness of existing solvers. On average, proposed methods significantly increase the state-of-the-art results by over five percentage points on benchmark datasets. Further, the solvers trained on the augmented dataset performs comparatively better on the challenge test set. We also show the effectiveness of proposed techniques through ablation studies and verify the quality of augmented samples through human evaluation. Future works could focus on developing techniques to generate data artificially and making robust MWP solvers.

Augmentation Method	**MaWPS**		**ASDiv-A**	
	Scratch	BERT	Scratch	BERT
True	77.2	**84.6**	53.2	**70.6**
Paraphrasing	79.8	**88.3**	58.1	**75.6**
Substitution	81.3	**89.2**	57.3	**73.2**
Combined	82.7	**91.3**	60.4	**78.2**

Table 4.11: Performance comparison of baseline model trained from scratch and trained using BERT embeddings. True represents unaugmented dataset.

Evaluation Criteria	**MaWPS**		**ASDiv-A**	
	Para	Sub	Para	Sub
Preserves Equation	92.3%	89.5%	93.6%	90.1%
Preserves Numbers	88.4%	91.2%	87.3%	90.3%
Semantic Similarity	0.96	0.89	0.91	0.87
Syntactic Similarity	4.67	4.36	4.59	4.33

Table 4.12: Human Evaluation scores on augmentated dataset. Para and Sub represents paraphrasing and substitution methods respectively.

Chapter 5

Conclusion and Future Works

The proposed methods and experiments in this work showcase that the existing math word solvers do not understand a math problem statement entirely and they are not robust enough for practical purposes. We devised automated adversarial attacks of high quality by utilizing and adjusting several key NLP techniques to fit our domain. Performance of existing solvers on these attacks showed a massive drop of 40 percent points. To understand the causes, why a solver is unable to perform well on the adversarial attacks, we conduct some experiments and based on the results of these experiments we hypothesize that the solvers do not pay heed to the question text, they are over-fitting to text in the vicinity of numerical quantities, and they are unable to generate new templates if the structure of the problem statement is modified. To counter this issue, we introduced several data augmentation techniques which focus on enhancing the diversity and overall size of the training dataset. These augmentations are curated automatically by ensuring that domain constraints are preserved. By performing training on the augmented datasets we observe an overall increase in the performance of existing solvers by 5 percent points on average. Also, on average the performance has now improved by 25 percent points on adversarial attacks.

With the advent of pretrained models, we believe that solvers will now be able to tackle word problems of higher complexities as well [11]. However, developing a fine tune dataset of considerable size is still a challenging task. Some of the approaches that can be used is by focusing on generating diversity rewrites for existing problem statements and ensuring that the solution expression still remains same. Further, we should also have a closer look at similarity measures for comparing two word problems. Another major area of focus could be on developing a unified metric which can rate the solution expressions generated by the solvers. We should induce mechanisms which can differentiate between a partially incorrect generation and mathematically incorrect generations as well.

To conclude this work tackles the task of enhancing the diversity and data size of math word datasets. The focus is to develop more robust and generalized math word solvers.

Related Publications

1. **Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2022.** Practice Makes a Solver Perfect: Data Augmentation for Math Word Problem Solvers. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4194–4206, Seattle, United States. Association for Computational Linguistics.
[Oral Presentation, NAACL 2022]

2. **Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021.** Adversarial Examples for Evaluating Math Word Problem Solvers. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2705–2712, Punta Cana, Dominican Republic. Association for Computational Linguistics. **[Poster Presentation, EMNLP 2021]**

Bibliography

[1] M. Aiken. The efficacy of round-trip translation for mt evaluation. 14, 02 2010.

[2] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

[3] A. Amini, S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019.

[4] A. Anastasopoulos, A. Lui, T. Q. Nguyen, and D. Chiang. Neural machine translation of text from non-native speakers. 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 3070–3080, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[5] M. Bayer, M.-A. Kaufhold, and C. Reuter. A survey on data augmentation for text classification. *ACM Computing Surveys*, 55(7):1–39, dec 2022.

[6] Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation, 2018.

[7] D. Bobrow. Natural language input for a computer problem solving system. 01 1964.

[8] E. Charniak. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, page 303–316, San Francisco, CA, USA, 1969. Morgan Kaufmann Publishers Inc.

[9] Y. Cheng, L. Jiang, and W. Macherey. Robust neural machine translation with doubly adversarial inputs. *arXiv preprint arXiv:1906.02443*, 2019.

[10] T.-R. Chiang and Y.-N. Chen. Semantically-aligned equation generation for solving and reasoning math word problems. *arXiv preprint arXiv:1811.00720*, 2018.

[11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways, 2022.

[12] V. Claveau, A. Chaffin, and E. Kijak. Generating artificial texts as substitution or complement of training data, 2021.

[13] C. Coulombe. Text data augmentation made simple by leveraging nlp cloud apis, 2018.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[15] E. Dinan, S. Humeau, B. Chintagunta, and J. Weston. Build it break it fix it for dialogue safety: Robustness from adversarial human attack. *arXiv preprint arXiv:1908.06083*, 2019.

[16] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[17] E. A. Feigenbaum and J. Feldman. *Computers And Thought*. Literary Licensing, LLC, Whitefish, MT, USA, 2012.

[18] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online, Aug. 2021. Association for Computational Linguistics.

[19] J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

[20] M. Glockner, V. Shwartz, and Y. Goldberg. Breaking nli systems with sentences that require simple lexical inferences. *arXiv preprint arXiv:1805.02266*, 2018.

[21] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith. Annotation artifacts in natural language inference data, 2018.

[22] Z. S. Harris. Distributional Structure. *WORD*, 10(2-3):146–162, Aug. 1954.

[23] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. 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, Oct. 2014. Association for Computational Linguistics.

[24] D. Huang, S. Shi, C.-Y. Lin, J. Yin, and W.-Y. Ma. 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, Aug. 2016. Association for Computational Linguistics.

[25] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[26] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits. 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, 2020.

[27] T. Jung, D. Kang, L. Mentch, and E. Hovy. Earlier isn't always better: Sub-aspect analysis on corpus and system biases in summarization. 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 3324–3335, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[28] V. Karpukhin, O. Levy, J. Eisenstein, and M. Ghazvininejad. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[29] D. Kaushik, E. Hovy, and Z. C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data, 2019.

[30] Y. Kim, P. Petrov, P. Petrushkov, S. Khadivi, and H. Ney. Pivot-based transfer learning for neural machine translation between non-English languages. 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 866–876, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[31] S. Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations, 2018.

[32] O. Kolomiyets, S. Bethard, and M.-F. Moens. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 271–276, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[33] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi. 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, 2016.

[34] W. Kryscinski, N. S. Keskar, B. McCann, C. Xiong, and R. Socher. Neural text summarization: A critical evaluation. 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 540–551, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[35] A. Kumar, S. Bhattamishra, M. Bhandari, and P. Talukdar. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. 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 3609–3619, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[36] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. 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, June 2014. Association for Computational Linguistics.

[37] D. Liu, Y. Gong, J. Fu, Y. Yan, J. Chen, J. Lv, N. Duan, and M. Zhou. Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5798–5810, Online, Nov. 2020. Association for Computational Linguistics.

[38] S. Longpre, Y. Wang, and C. DuBois. How effective is task-agnostic data augmentation for pretrained transformers?, 2020.

[39] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[40] R. Maheshwary, S. Maheshwary, and V. Pudi. A context aware approach for generating natural language attacks. *arXiv preprint arXiv:2012.13339*, 2020.

[41] R. Maheshwary, S. Maheshwary, and V. Pudi. Generating natural language attacks in a hard label black box setting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.

[42] N. Malandrakis, M. Shen, A. Goyal, S. Gao, A. Sethi, and A. Metallinou. Controlled text generation for data augmentation in intelligent artificial agents. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 90–98, Hong Kong, Nov. 2019. Association for Computational Linguistics.

[43] J. Mallinson, R. Sennrich, and M. Lapata. 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, Apr. 2017. Association for Computational Linguistics.

[44] V. Marivate and T. Sefara. Improving short text classification through global augmentation methods. In *Lecture Notes in Computer Science*, pages 385–399. Springer International Publishing, 2020.

[45] R. T. McCoy, E. Pavlick, and T. Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

[46] S.-y. Miao, C.-C. Liang, and K.-Y. Su. 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, July 2020. Association for Computational Linguistics.

[47] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3(4):235–244, 12 1990.

[48] J. Miller, K. Krauth, B. Recht, and L. Schmidt. The effect of natural distribution shift on question answering models, 2020.

[49] A. Naik, A. Ravichander, N. Sadeh, C. Rose, and G. Neubig. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[50] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[51] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, Nov. 2016. Association for Computational Linguistics.

[52] A. Patel, S. Bhattamishra, and N. Goyal. 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, June 2021. Association for Computational Linguistics.

[53] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[54] A. Poliak, J. Naradowsky, A. Haldar, R. Rudinger, and B. Van Durme. Hypothesis only baselines in natural language inference, 2018.

[55] S. Qiu, B. Xu, J. Zhang, Y. Wang, X. Shen, G. de Melo, C. Long, and X. Li. Easyaug: An automatic textual data augmentation platform for classification tasks. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 249–252, New York, NY, USA, 2020. Association for Computing Machinery.

[56] E. Rabinovich, R. N. Patel, S. Mirkin, L. Specia, and S. Wintner. Personalized machine translation: Preserving original author traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1074–1084, Valencia, Spain, Apr. 2017. Association for Computational Linguistics.

[57] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[58] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[59] G. Rizos, K. Hemker, and B. Schuller. Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[60] D. Saxton, E. Grefenstette, F. Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

[61] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data, 2016.

[62] H. Shi, K. Livescu, and K. Gimpel. Substructure substitution: Structured data augmentation for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3494–3508, Online, Aug. 2021. Association for Computational Linguistics.

[63] J. F. Steffensen. *Interpolation / J. F. Steffensen*. London, 1927.

[64] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[65] H. Tan, L. Yu, and M. Bansal. Learning to navigate unseen environments: Back translation with environmental dropout, 2019.

[66] L. Taylor and G. Nitschke. Improving deep learning with generic data augmentation. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1542–1547, 2018.

[67] V. Vaibhav, S. Singh, C. Stewart, and G. Neubig. Improving robustness of machine translation with synthetic noise. 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 1916–1920, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[68] E. Voita, R. Sennrich, and I. Titov. Context-aware monolingual repair for neural machine translation, 2019.

[69] Z. Wan, X. Wan, and W. Wang. Improving grammatical error correction with data augmentation by editing latent representation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2202–2212, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics.

[70] H. Wang, P. Zhang, and E. P. Xing. Word shape matters: Robust machine translation with visual embedding, 2020.

[71] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. Dai, and H. Shen. Template-based math word problem solvers with recursive neural networks. In *AAAI*, 2019.

[72] Y. Wang, X. Liu, and S. Shi. 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, Sept. 2017. Association for Computational Linguistics.

[73] J. Wei and K. Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. 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 6382–6388, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[74] S. Witteveen and M. Andrews. Paraphrasing with large language models. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 215–220, Hong Kong, Nov. 2019. Association for Computational Linguistics.

[75] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[76] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, and Q. V. Le. Unsupervised data augmentation for consistency training, 2020.

[77] Z. Xie and S. Sun. 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, 7 2019.

[78] Z. Xie and S. Sun. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305, 2019.

[79] Z. Xie, S. I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, and A. Y. Ng. Data noising as smoothing in neural network language models, 2017.

[80] M. Xu, S. Yoon, A. Fuentes, and D. S. Park. A comprehensive survey of image augmentation techniques for deep learning, 2022.

[81] Y. Yang, C. Malaviya, J. Fernandez, S. Swayamdipta, R. Le Bras, J.-P. Wang, C. Bhagavatula, Y. Choi, and D. Downey. Generative data augmentation for commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1008–1025, Online, Nov. 2020. Association for Computational Linguistics.

[82] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim. 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, July 2020. Association for Computational Linguistics.

[83] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

[84] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification, 2016.

[85] G. G. Şahin and M. Steedman. Data augmentation via dependency tree morphing for low-resource languages, 2019.