Numerite: An Automatic Math Word Problem Solver

Anonymous RANLP 2023 submission

Abstract

In this paper, we present *Numerite*, an automatic math word problem solver that competes with previous solutions for solving elementary math word problems. Automatic math word problem solving is a complex multi-disciplinary task that involves the application of *Natural Language Processing and Understanding*, *Deep Learning*, as well as commonsense world knowledge. Numerite uses a BERT-based operation classifier and a rule-based context identifier to cohesively form an equation that can solve math word problems on elementary-level single-operator arithmetic word problems involving addition, multiplication, subtraction and division. Numerite is trained and tested on notable datasets in the field of automated math word problems solving.

1 Introduction

A Math Word Problem (MWP) is a mathematical exercise that exclusively uses natural language to introduce the premise of the problem with relevant background information and pose a question based on it, instead of using mathematical notations to pose the problem. Commonly used languages like English, Hindi, Spanish or any other language qualify as natural language. In our research, we focus solely on math word problems in English that are of a single variable and at the primary school level. The Automatic MWP Solving task involves semantic parsing, wherein the objective is to produce an equation that correctly represents the MWP, that can be assessed to obtain the solution. (Patel et al. (2021)) Automatic solving of MWPs is a task that involves processing the natural language of the problem semantically, with the end goal of generating an equation that correctly represents the MWP, which can then be solved to obtain the answer.

The complexity of the task of automatic MWP solving can be attributed to two major factors: the need for ample domain knowledge to have context, and linguistic understanding to gain meaning. In other words, there exists a large semantic gap (Zhang et al. (2020a)) between human-readable words and machine-understandable logic. (Patel et al. (2021)) provide evidence to show that several benchmark models have used shallow heuristics for performance measurement, rendering their results unreliable. They find that these models are able to achieve high accuracy without even considering the question, solely relying on background world information in the MWP. Such models exploit the superficial patterns in most MWPs in order to solve them, as opposed to solid logic. This proves that there is still much work to be done for basic arithmetic MWP solving, before moving on to more complex problems.

MWPs are diverse, ranging from single unknown variable problems at the primary school level, to multiple unknowns at the collegiate level. They can cover a wide range of mathematical concepts, from calculus to geometry to arithmetic to trigonometry. There is also a wide variety of operations possible, the most basic ones being the arithmetic operators (+, -, /, *) and the complex ones including calculus, and other complicated concepts. Our decision to select single variable simple arithmetic word problems with only 4 possible operations stems from the fact that current models simply do not have enough accuracy to regard this problem as solved. We think that we must work on and master the basics (solving simple primary school level MWPs) before research and work is done on more complex MWPs (geometry, trigonometry, algebraic, calculus, probability and statistics MWPs).

In this paper, we present Numerite, an automatic word problem solver that follows a hybrid-modular approach, which utilizes a combination of rule-based and deep learning techniques. Information from several intermediate modules is combined to generate the correct mathematical equation for the word problem posed. The intermediate tasks are split up as: operation identification, microstatements generation, question statement identification, knowledge base creation, and irrelevant information removal (Mandal et al. (2020)). Finally, all the transitional outputs from these modules are sent to an equation generation unit, that generates the final equation.

2 Related Works

Automatic math word problem-solving has been garnering research interest over the years. So far, research has focused on solving various kinds of math word problems including trigonometric, multi-variate, geometric and arithmetic word problems. Various deep-learning techniques have been explored and many datasets have been developed for this. Several surveys have also assessed existing solutions for automatic math word problem solvers and datasets that have been published.

One such survey was conducted by Sundaram et al. (2022) which compares various MWP solvers and their performances on various datasets and reveals a shocking truth: MWP solvers still struggled when it came to solving elementary math problems, in spite of extensive research. This survey exposed that we have moved on to solving more complex word problems, even without mastering simple word problems, and under the assumption that current solvers were excellent at solving elementary math word problems.

Another thoroughgoing paper by the makers of SVAMP (Patel et al. (2021)) exposes solvers built on benchmark datasets. First, they expose that the majority of problems in benchmark datasets can be solved by shallow heuristics lacking word-order information or lacking question text. Second, they present a challenge set (SVAMP) for better evaluation of methods. They explain deficiencies in existing datasets. Analyzing the large gap in results on benchmark datasets and the SVAMP dataset, they sufficiently show that the capability of existing models to solve simple one-unknown arithmetic word problems is overestimated. Other papers by Zhang et al. (2020a) and the creators of the Dolphin18k dataset (Huang et al. (2016)) reveal that when models were applied on large and diverse datasets, none of the proposed methods in the literature achieves high precision, revealing that current MWP solvers still have much room for improvement. Using these as a reference, we can comfortably conclude that the problem of automatic MWP solving for even elementary problems is still far from being solved.

According to Zhang et al. (2020a), researchers have attempted to solve this problem using rule-based methods (1960-2010), using Feature Engineering, Statistical Learning and Semantic Parsing (2011-2017) and finally, deep-learning and reinforcement learning methods (2018 onwards).

Some methods, implemented by Hosseini et al. (2014) involve Verb Categorization, where the problem statements are grounded to a whole 'world state transition' by combining representations of individual statements. Others make use of tree-based algorithms, like the tree-structured decoding algorithm that generates a top-down expression tree for the word problem (Liu et al. (2019)), translating the MWP to a Universal Expression Tree that can be easily solved (Wang et al. (2018)), or graph-based methods like (Zhang et al. (2020b)) that effectively represent orders and quantities of MWPs in order to solve them. Qin et al. (2020) adds on these methods and introduces a Semantically-Aligned Universal Tree-Structured Solver.

Another method of Neural-Symbolic Computing was implemented by (Jinghui et al. (2021)) where a problem reader is used to encode the math word problems into vector representations. Four novel tasks: common sense checks, number predictors, program consistency checkers and a duality exploiting task were introduced by this paper.

On the flip side, more and more research has focused on the use of seq2seq models by Huang et al. (2018) and RNN-based architectures with attention by Wang et al. (2017). Several researchers have also made use of LSTM and Bi-LSTM. Most notably, Mandala et al.,2021 (Mandal et al. (2020)) make use of an RNN-based BiLSTM network to classify operations. They also developed a Knowledge-Based IIRU, a relevance classifier, to identify the relevant quantities (RQ) required to solve MWPs.

Some recent research also made use of pre-trained transformers such as BERT (Devlin et al. (2019)) for language understanding and taking a more syntactical approach to the problem. Liang et al. (2021) attempts to solve the problem statement by making use of BERT to create a contextualized learning schema and inject numerical properties into the symbolic placeholders.

Our work makes uses of classic NLP techniques as presented in Mandal et al. (2020), Hosseini et al. (2014), as well as Deep Learning techniques presented in Devlin et al. (2019), Huang et al. (2018). The datasets were also chosen taking into consideration the points made in Patel et al. (2021) and Zhang et al. (2020a).

3 Data

The most suited data for this research were those that contain single-operator double-operand math word problems, along with their solutions, and the operator used to solve the problem. Renowned datasets like Dolphin18k and Math23k (Chinese) were not considered because of the increased complexity of the problem and language barriers, respectively. The datasets chosen for the problem statement were SingleOp and SVAMP (Simple Variations on Arithmetic Math Problems) dataset.

The concept of the dataset "SingleOp" was theorized and implemented in Roy and Roth (2015). The problems were collected from real-world websites that are used by primary school children to practise word problems. The data was checked and cleansed of any problems that required background knowledge (a week consists of 7 days, a dozen consists of 12 units, etc.). This left the authors of the dataset with a collection of single-operation arithmetic word problems. The problems in the dataset however lacked any variability in linguistic terms. Problems solved using a certain operator were usually framed in a similar manner.

To alleviate the problem of datasets with shallow patterns, Patel et al. (2021) presents a new dataset called "SVAMP" (Simple Variations on Arithmetic Math word Problems). The development of SVAMP was motivated by the need to address the issue of limited variability in datasets such as AsDiv-A. While AsDiv-A shares similarities with other datasets in terms of scope and difficulty, SVAMP presents a unique challenge by reducing the susceptibility of models to rely on superficial patterns for successful solutions.

Our gathered data consists of single variable Math Word Problems of the primary school level, with the 4 possible mathematical operations being addition, multiplication, division or subtraction.

3.1 Features

The SingleOp dataset contains 562 rows and 3 relevant columns. The columns include - lEquations - contains the solvable equation in a string lSolutions - contains the final answer, enclosed in a list sQuestions - contains the Math Word Problem in its entirety

The SVAMP dataset contains 1000 rows and 6 relevant columns. The columns include - Question - contains the Math Word Problem in its entirety; however, all numerical values are replaced with a reference to them. Numbers - contains the actual values that are being referenced in the “Question” column Equation - contains the prefix equation with reference numbers Answer - contains the final numerical solution of the word problem Body - contains just the factual portion of the MWP Ques - contains just the question goal of the MWP

3.2 Preprocessing

The datasets did not require extensive preprocessing, with only a few preprocessing steps doing the job.

For the SingleOp dataset, preprocessing involved deriving a column "Type" from "lEquations", to keep track of the operation being performed in each dataset, and rows containing duplicate data were removed. After preprocessing, 531 unique word problems remained.

In the SVAMP dataset, the “Numbers” column was used to replace the references in the other columns and was then dropped. The equation was converted from prefix to infix form to allow for better readability. It was further used to create a “Type” column which specified the operation being performed (Addition, Subtraction, Multiplication, Division). The columns of “Body” and “Ques” were dropped after they had been sufficiently utilized to solve the sub-problem of Question Identification. Rows containing word problems that required more than a single operator to solve were collected and removed from the usable dataset. The number of samples that remained after preprocessing is 762.

3.3 Data distribution

The preprocessed datasets were further analysed to discover the distribution of the data in terms of the operations being performed in each of the word problems. The SingleOp dataset consisting of 531 different word problems contained 154 problems of addition, 162 problems of subtraction, 113 problems of multiplication and 102 problems of division. The SVAMP dataset consisting of 762 different word problems contained 139 problems

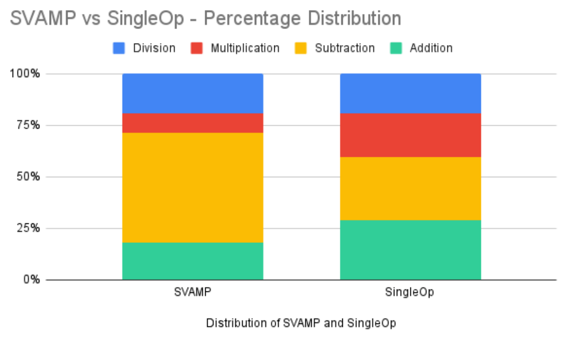

Figure 1: SVAMP vs SingleOp Data Distribution

of addition, 404 problems of subtraction, 74 problems of multiplication and 145 problems of division. There was a heavy skew in the data towards Subtraction, with more than 50% of the questions belonging to that label.

Compared to the SVAMP dataset, the SingleOp dataset was more consistent in its distribution.

4 Methodology

Numerite has been implemented using a hybrid-modular approach, which utilizes a combination of rule-based and deep learning techniques for NLP tasks needed to build Numerite. Our methodology will be clearly detailed in the following sections. We will use the following sample word problem across all sections of the methodology to clearly illustrate the methodology of Numerite over the course of this text. — ***"Peter has sixteen eggs and 8 balloons. If he shares the eggs among 4 friends, how many eggs does each friend get?"***

Numerite performs several sub-tasks whose intermediate outputs are essential for generating a final mathematical equation for the word problem supplied. The most notable tasks are Operation Identification, Microstatements Generation, Question Identification, Knowledge Base Generation, Removal of Irrelevant Information, and finally, the Equation Generator and Solver.

4.1 Operator Identification

We model the task of operation identification of the MWP as a multi-class text classification task, with the only four classes being: Addition, Subtraction, Multiplication, and Division. We have used deep learning using BERT (Devlin et al. (2019)) to classify the word problem. We use a combination of the SingleOp and SVAMP datasets for training and validation purposes. Each MWP from the combined dataset is preprocessed to make it compatible with training, and the target class labels are encoded. Then, the pre-trained BERT model is loaded and further trained for hours on the training set of our preprocessed combined dataset. While predicting the operation for a given input MWP, the MWP needs to be preprocessed similar to the problems the trained BERT model was trained on. It outputs the tag associated with the label, which is then decoded to the final predicted operation. It takes as input a raw MWP in natural language and outputs an arithmetic operation among Addition, Subtraction, Multiplication, and Division: that the MWP is most likely to be.

For the sample problem - *"Peter has sixteen eggs and 8 balloons. If he shares the eggs among 4 friends, how many eggs does each friend get?"* The predicted operation will be ***division***.

4.2 Microstatements Generation

4.2.1 Pre-Processing

All the characters in the MWP are converted to lowercase characters for consistency. Using the free version of 'JamSpell' [1], a Python package, all spelling errors are rectified. For correction of punctuation errors in the MWP, we use the Python package 'deepmultilingualpunctuation'. In addition, Any number names present are converted back to their numeric form using a combination of rule-based techniques and the use of the word2number package. In the sample example, "sixteen" will be converted to the numerical "16". Then, in case there exists a stranded number without a noun directly associated with it, it must be followed by the noun it refers to. For example, in the phrase, *"John bought 3 and Amy bought 4 marbles..."*, the quantity '3' is stranded, so we will append the noun 'marbles' to the quantity to make our downstream task clearer. We get, *"John bought 3* ***marbles*** *and Amy bought 4 red marbles..."* as a result.

Finally, It refers to the task of finding and replacing pronouns with their respective noun phrases. This task is performed to remove any ambiguity that can occur in individual microstatements. Since each microstatements will contribute to the knowledge base independently of the other, it is important for each of them to have complete information of the world knowledge it represents. This task is performed by the neuralcoref package [2], which is

[1] Jamspell: A spelling correction module for Python. `https://jamspell.com`.

[2] NeuralCoref: A coreference resolution module based

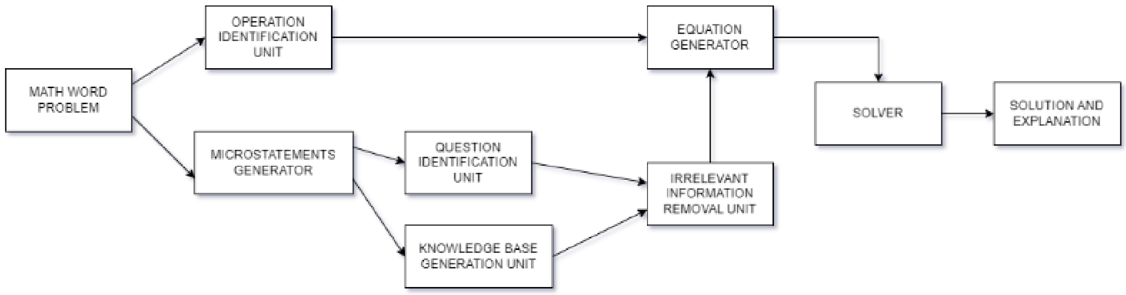
MATH WORD PROBLEM
OPERATION IDENTIFICATION UNIT
MICROSTATEMENTS GENERATOR
QUESTION IDENTIFICATION UNIT
KNOWLEDGE BASE GENERATION UNIT
IRRELEVANT INFORMATION REMOVAL UNIT
EQUATION GENERATOR
SOLVER
SOLUTION AND EXPLANATION

Figure 2: Methodology of Numerite

a state-of-the-art coreference resolution based on neural nets and spaCy. For example, the question - "Peter has sixteen eggs and 8 balloons. If *he* shares the eggs among 4 friends, how many eggs does each friend get?" will be converted to "Peter has sixteen eggs and 8 balloons. If *Peter* shares the eggs among 4 friends, how many eggs does each friend get?"

Output Microstatements generated by this unit for the sample are: ['peter has 16 eggs', 'peter has 8 balloons', 'if peter shares the eggs among 4 friends', 'how many eggs does each friend get']

4.3 Question Identification

The task of question identification involves distinguishing between the microstatements that establish facts in the MWP, versus the microstatements that define the question goal of the MWP. The information gained by this task can be further used in the Irrelevant Information Removal Module. Though most questions have the question goal at the very end, it is not always guaranteed. In the case of MWPs such as "How many eggs does Peter have left, given that Peter had 4 eggs, out of which he ate 2 eggs?", the assumption that the question goal follows the fact-based microstatements will lead to incorrect answers.

The input to this module is the list of microstatements generated, and the output is a dictionary with the labels - "statements" and "question". Since each microstatement could be either a statement or a question, the task of question identification can be reduced to that of a Binary Classification. The task was approached using multiple methods, including Deep Learning Binary Cross-entropy, Rule-Based Identification and Logistic Regression. However, the rule-based approach proved to be the best fit.

in spaCy https://github.com/huggingface/neuralcoref.

To create the Rule-Based Approach, SVAMP (Patel et al. (2021)), SingleOp (Roy and Roth (2015)) and Dolphin18k[14] datasets were used. Questions and statements were manually identified and separated from the MWP, and checked for any inconsistencies. The word count was calculated on the dataset containing questions. This allowed us to obtain a list of the most commonly occurring words in a question. Similarly, a list of words occurring in statements was found. A difference operation between the two lists was performed to achieve a list of words that are exclusive to questions. These words include— "evaluate", "calculate", "how", "determine", "find", "what", and "identify". Each microstatement is checked for the occurrence of one of these words and is determined as a question or statement based on that examination, and added to the relevant dictionary.

For the sample MWP, the question will be identified as "How many eggs does each friend get?", and the rest of the microstatements will be considered as world state information statements.

4.4 Knowledge Base Generation

After separating the microstatements into question and non-question 'world state' microstatements, the task is to extract all possible information from them. Nouns, adjectives and cardinals are extracted from each microstatement and made to associate with its WorldStateKB set.

This is accomplished by first lemmatizing the MWP with the WordNetLemmatizer from nltk. Then, using the inflect python module, all nouns are converted to their plural form. All adjectives and numbers are also extracted. Finally, we get a key-value pair of microstatements numbered from 1 to n and the knowledge set associated with that particular microstatement. We also create a cardinals dictionary, with key-value pairs of microstatement numbers and the quantities associated with them.

Microstatements for world state information of the sample MWP: ['peter has 16 eggs', 'peter has 8 balloons', 'if peter shares the eggs among 4 friends'], and for question: ['how many eggs does each friend get']

The knowledge base generation for the sample MWP will yield:

Knowledge Base for Questions: 1: 'eggs'

Knowledge Base for World State: 1: 'eggs', 'peter', 2: 'balloons', 'peter', 3: 'shares', 'friends', 'peter',

World State Cardinals 1: [16.0], 2: [8.0], 3: [4.0]

At the end of this step, we have successfully extracted the cardinals associated with each element of the knowledge base of the world state.

4.5 Removal of Irrelevant Information

This task is extremely important to find all the relevant microstatements and the quantities associated with them that will be present in the final equation. To identify irrelevant information, we will need the knowledge base associated with the question microstatements and the world state microstatements each.

The logic behind identifying irrelevant microstatements is inspired by the method proposed by Mandal et al. (2020). Before this process, all the keywords in each knowledge base must be converted to sets. A statement is classified as relevant if all the information from the question is present in the microstatement.

To find irrelevant microstatements, the process is as follows:

1. Compute the cue differences between the knowledge base set of the question statements (QuestionKB) and the knowledge base for each microstatement n from the set of world state microstatements (WorldStateKB(n)).

   The result is computed as follows: result(n) = QuestionKB - (QuestionKB WorldStateKB(n))

2. If the result of the cue difference is a null set (ϕ), then we consider the microstatement associated with that set to be relevant. If the result is a non-null set, it is discarded and marked as irrelevant information.

   If result(n) = ϕ, then the microstatement n is marked relevant and the quantity associated with it is retained in the WorldStateKB. Else, if result(n) $\neq \phi$, then the microstatement n is marked irrelevant and its quantity is discarded from the WorldStateKB.

4.6 Equation Generation

Numerite generates a correct equation for the MWP using all the information gained so far. After removing all irrelevant world state info, only relevant quantities remain in the knowledge base. The relevant quantities are extracted as the operands of the final equation. Using these operands, and the operator that we predicted using Operation Identification, we can generate the equation for this word problem.

The solution of an equation may depend on the order of the operands. The commutative property states that a function is commutative if, $f(a, b) = f(b, a)$. If the operation performed in the given equation is Commutative (Addition or Multiplication), the order of the operands does not impact the true meaning of the equation and its solution. Unlike Addition and Multiplication, Subtraction and Division are not commutative. This means that the order of the operands will fundamentally change the meaning of the equation and produce different solutions. To account for this order, we assume that at a primary school level, arithmetic word problems do not deal with negatives and/or fractions, and therefore the higher-valued operand will be placed before the lower-valued operand.

4.7 Solving the Equation

Numerite is able to generate a step-by-step solution to the MWPs it solves, providing the following as output:

- The operation required to solve the MWP
- The final equation corresponding to the MWP
- The explanation of the MWP

5 Results and Discussion

Numerite is built to solve MWPs that contain exactly one operation among addition, division, subtraction and multiplication. If we are able to correctly identify the operation expected from a math word problem, and all the relevant operands (quantities) needed, we can be confident about framing the correct equation for that MWP.

We test the accuracy of Numerite on the SingleOp Roy and Roth (2015) and SVAMP Patel et al. (2021) datasets and explore the impact of

Dataset	Add	Sub	Mul	Div
SingleOp	96.1%	85.8%	78.8%	92.2%
SVAMP	69.1%	95.1%	68.9%	89.6%

Table 1: Accuracy of the Operation Identification Task

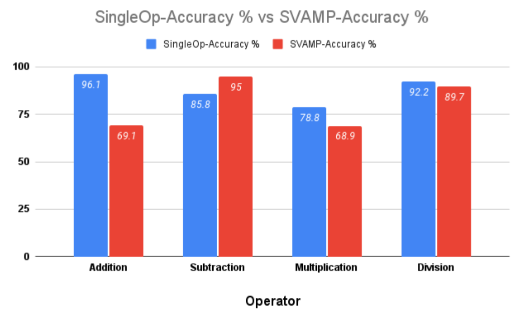

Figure 3: Operator-wise Accuracy of Numerite

Irrelevant Information Removal Unit and Operator Identification on the final accuracy of solving MWPs.

We use the SingleOp and SVAMP datasets to test the accuracy of Numerite, for each type of operation. We also measure the overall accuracy of Numerite across classes of operations.

Operation Identification is the only task in our algorithm that runs independently of the other tasks involved in the problem-solving process. Every word problem has a single operator associated with it. The correct identification of this operator dictates half of the success of the MWP solving task, in short: "the task of math word problem solving can only be as good as the task of operator identifying". Therefore, we deem this a necessary statistic to calculate.

The overall accuracy of the Operation Identification task on both datasets is **87.47%**. The high accuracy of Subtraction in SVAMP is caused due to the presence of a high volume of subtraction questions.

The final accuracy of the system on the SingleOp dataset is **77.21%**, while the final accuracy on the SVAMP dataset is **48.43%**. For example, if Numerite is unable to correctly identify the operation of an MWP, it will not be able to generate a correct equation despite correctly identifying relevant quantities. Similarly, if the operation is correctly predicted but the relevant quantities extracted are wrong, then we will still end up with an incorrect equation. These errors can stem from upstream tasks like Operation Identification, Microstatements Generation, and Irrelevant Information Removal.

Dataset	Add	Sub	Mul	Div
SingleOp	73.4%	66.7%	63.7%	60.8%
SVAMP	22.3%	39.4%	29.7%	46.2%

Table 2: Accuracy drop after bypassing IIRU

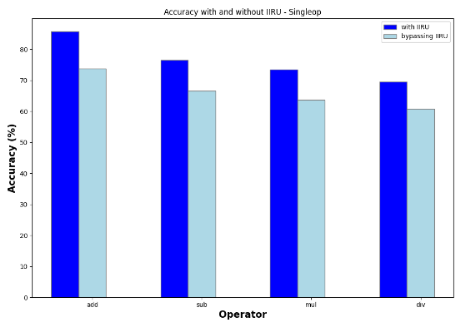

Figure 4: Accuracy Drop After Bypassing IIRU

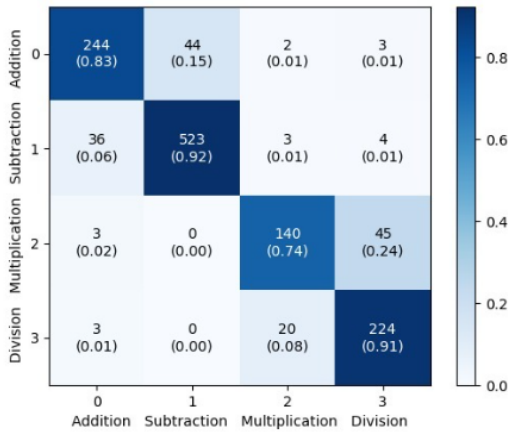

Figure 5: Confusion Matrix for the Operation Prediction Task

The comparatively low accuracies from running on the SVAMP dataset can be attributed to the fact that SVAMP contains a number of MWPs with punctuation errors, where commas are missing from a few MWPs. The accuracies of SingleOp and SVAMP are heavily affected by the fact that minor issues in upstream tasks (issues like missing punctuation) can cause cascading errors, and hugely increase the chances of making mistakes in downstream tasks. Another reason for low accuracies on SVAMP is that the dataset is specifically built to challenge even the most robust state-of-the-art models by including simple, yet unexpected variations in standard math word problems.

However, even lower accuracies on SVAMP are

Dataset	Accuracy
SingleOp	77.21%
SVAMP	48.43%

Table 3: Overall accuracy of Numerite

Model	SingleOp	SVAMP
Roy et. al	73.9%	-
GTS	-	31.7%
Graph2Tree	-	42.9%
Seq2Seq	-	25.04%
Numerite	77.21%	**48.43%**

Table 4: Comparison with benchmarks

more valuable than higher accuracy models on SingleOp, solely because SVAMP is intended to be a tricky dataset, without the possibility of models exploiting shallow patterns and heuristics— it is specifically built to challenge even the most robust state of the art models by including simple, yet unexpected variations in standard math word problems.

Our model, Numerite, performs better than the SOTA approaches by at least 5% on the SVAMP dataset. This can be credited to the unique approach to the nature of the problem. By emphasizing the importance of identifying relevant information and the operation involved in the question, we can achieve higher accuracy.

6 Conclusion

Our hybrid approach of deep learning and rule-based models in building Numerite successfully solves elementary-level word problems of a single operation with two operands, given some defined entities, owners, and quantities. By achieving a final accuracy of 77.21% on the SingleOp dataset (Roy and Roth (2015)) and 48.43% on the SVAMP (Patel et al. (2021)) dataset, we surpass state-of-the-art (SOTA) performance on these datasets, as shown in Table 4. Our approach uses a trained BERT model for operation prediction and employs various linguistic and rule-based approaches to identify and eliminate irrelevant quantities. The Irrelevant Information Remover Unit (Mandal et al. (2020)) implementation evidences the importance of identifying relevant quantities and the correct operation to accurately solve math word problems of this nature.

Our unique approach enables us to validate and verify the system's performance at each critical step, which can explain the solution to the math word problem. Moreover, our approach generates readable output at every step of the solving process, making it an excellent tool for primary to middle school students. Despite the seeming simplicity of solving math word problems, our project exposed the complexities involved in crafting and dissecting a word problem, even for the most straightforward cases, requiring attention to detail. Our project results demonstrate promise for further contributions to this field of Natural Language Processing.

7 Future Scope

While Numerite is already able to successfully solve a sizable amount of single-operation math word problems automatically, automatic math word problem solvers still have a long way to go.

In the future, using even larger amounts of data, with higher variability in terms of linguistic structure, would most definitely allow us to pursue harder math word problems, and produce an overall higher accuracy. We find the need to develop a more balanced MWP dataset.

Additionally, we think a more rule-based module would also have a positive impact on the performance of our model. We would like to experiment with employing a stronger parser and a stronger co-reference resolver.

This idea can be extended to solve harder word problems such as those involving multiple operators and operands, or more complex operations like percentages, ratios, profit-loss, calculus and more, by exploring different deep learning and rule-based approaches.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 887–896, Berlin, Germany. Association for Computational Linguistics.

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 419–428, Melbourne, Australia. Association for Computational Linguistics.

Qin Jinghui, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. pages 5870–5881.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2021. Mwp-bert: Numeracy-augmented pre-training for math word problem solving. In *NAACL-HLT*.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. 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 2370–2379, Hong Kong, China. Association for Computational Linguistics.

Sourav Mandal, Arif Ahmed Sekh, and Sudip Naskar. 2020. Solving arithmetic word problems: A deep learning based approach. *Journal of Intelligent Fuzzy Systems*, 39:1–11.

Arkil Patel, S. Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *North American Chapter of the Association for Computational Linguistics*.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3780–3789, Online. Association for Computational Linguistics.

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

Sowmya S Sundaram, Sairam Gurajada, Marco Fisichella, Deepak P, and Savitha Sam Abraham. 2022. Why are nlp models fumbling at elementary math? a survey of deep learning based word problem solvers.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069, Brussels, Belgium. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2020a. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2287–2305.

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