Solving Math Word Problem with Problem Type Classification

Jie Yao, Zihao Zhou, and Qiufeng Wang(✉)

School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou, China
{Jie.Yao22,Zihao.Zhou22}@student.xjtlu.edu.cn, Qiufeng.Wang@xjtlu.edu.cn

Abstract. Math word problems (MWPs) require analyzing text descriptions and generating mathematical equations to derive solutions. Existing works focus on solving MWPs with two types of solvers: tree-based solver and large language model (LLM) solver. However, these approaches always solve MWPs by a single solver, which will bring the following problems: (1) Single type of solver is hard to solve all types of MWPs well. (2) A single solver will result in poor performance due to over-fitting. To address these challenges, this paper utilizes multiple ensemble approaches to improve MWP-solving ability. Firstly, We propose a problem type classifier that combines the strengths of the tree-based solver and the LLM solver. This ensemble approach leverages their respective advantages and broadens the range of MWPs that can be solved. Furthermore, we also apply ensemble techniques to both tree-based solver and LLM solver to improve their performance. For the tree-based solver, we propose an ensemble learning framework based on ten-fold cross-validation and voting mechanism. In the LLM solver, we adopt self-consistency (SC) method to improve answer selection. Experimental results demonstrate the effectiveness of these ensemble approaches in enhancing MWP-solving ability. The comprehensive evaluation showcases improved performance, validating the advantages of our proposed approach. Our code is available at this url: https://github.com/zhouzihao501/NLPCC2023-Shared-Task3-ChineseMWP.

Keywords: Math Word Problem · Ensemble Learning · Bert2Tree · Large Language Model

1 Introduction

Math word problems (MWPs) are primarily solved by analyzing the text description of the problem and automatically generating mathematical equations to derive the solution, as illustrated in Table 1(a). Initially, the solver extracts the problem's text description and applies pre-processing techniques, including semantic parsing. Subsequently, leveraging the processed text description, the solver examines the mathematical logic relationships with the associated concepts and generates the relevant mathematical equations. Finally, by utilizing the generated equations, the solver obtains the corresponding answers.

J. Yao and Z. Zhou—Equal contribution.

F. Liu et al. (Eds.): NLPCC 2023, LNAI 14304, pp. 123–134, 2023.
https://doi.org/10.1007/978-3-031-44699-3_12

Table 1. Examples of math word problem

(a) General MWP:
Text: Dingding has read 180 pages of a book and has 150 pages left to read. How many pages are there in this book?
Equation: x = 180 + 150
Answer: 330
(b) Law Finding MWP:
Text: Find the pattern and fill in the numbers. 2, 6, 10, __ , 18
Equation: x = 14
Answer: 14
(c) Unit Conversion MWP:
Text: The ratio of bean paste to white sugar is 2:1. Now there are 450 grams of white sugar, how many kilograms of bean paste are needed?
Equation: x = 450 * 2 ÷ 1000
Answer: 0.9

In recent years, a multitude of natural language processing (NLP) techniques have emerged to tackle MWPs [1], encompassing advancements in semantic parsing and deep learning. Semantic parsing serves as a powerful approach to decompose the textual content of a math problem into structured representations, facilitating the generation of corresponding mathematical expressions [8,14]. Numerous methodologies have been proposed for semantic parsing, spanning rule-based and statistical methods. With the boom of deep learning, the research on solving MWPs has recently made great progress. For example, tree-based models [19,21] as well as large language models (LLM) [16,18,22] have been extensively exploited to deal with MWPs, and increase the accuracy of prediction significantly.

However, these approaches always solve MWPs by a single solver, which usually brings the following two problems. (1) Single type of solver is hard to solve all types of MWPs well. For example, the tree-based solver is unable to solve some types of MWPs like law finding problems (e.g., Table 1(b)) because it relies on combining numbers into MWP and operators (+-*/) to get an answer equation, while the LLM solver is unable to solve complex MWPs due to lacking calculation ability. (2) A single solver tends to result in poor performance due to over-fitting.

To address these challenges, we adopt the following two approaches. (1) To combine the abilities of the tree-based solver and the LLM solver, we propose a problem type classifier. Specifically, we define some heuristic rules to divide MWP types into two categories. One is for LLM solver such as law finding problems and unit conversions problems (e.g., Table 1(c)), and the other is for tree-based solver. (2) To avoid over-fitting and improve the performance of the

LLM solver and tree-based solver, we apply ensemble techniques to both of them. For the tree-based solver, we propose an ensemble learning framework based on ten-fold cross-validation and voting mechanism. In the LLM solver, we adopt the self-consistency (SC) method to select the most appropriate answer and enhance the model's overall performance. Figure 1 shows an overview of our method (Ensemble-MWP). Firstly, the problem type classifier assigns each MWP to one category. Then the corresponding solver (either the tree-based or LLM solver) will process the MWP to obtain a preliminary result. Lastly, we adopt a post-processing method to obtain the final answer. In summary, our contributions are as follows:

- We propose a problem type classifier to combine the abilities of both the tree-based solver and the LLM solver. To the best of our knowledge, this is the first effort to integrate them.
- We propose an ensemble learning framework based on ten-fold cross-validation and voting mechanism for the MWP solver.
- Experimental results demonstrate the effectiveness of these ensemble techniques in enhancing the ability to solve MWPs.

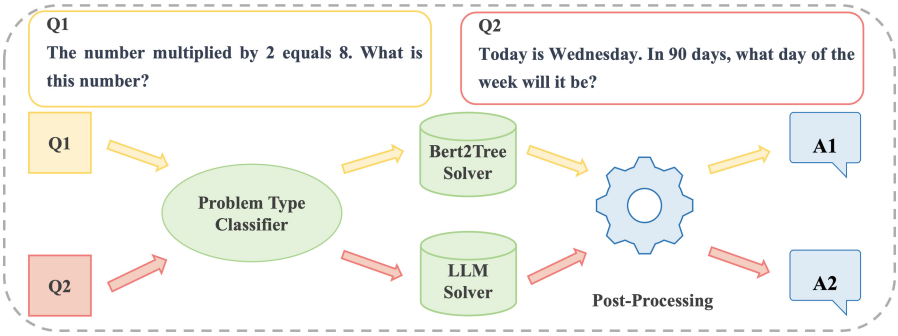

Fig. 1. Overview of Ensemble-MWP. The **Problem Type Classifier** assigns each MWP to either the **Bert2Tree solver** or the **LLM solver** based on a set of predefined rules. Once the classification is determined, the respective solver is employed to process the MWP and generate a preliminary result. The obtained result undergoes further **Post-processing** to derive the final answer

2 Related Work

2.1 Ensemble Learning

Ensemble learning has gained popularity for its ability to enhance predictive performance by combining multiple models. Bagging, a widely adopted ensemble learning technique, aims to reduce learner variation by training multiple

samples using the same learning algorithm. Lin [11] conducted a study demonstrating the effectiveness of bagging methods in improving the performance of NLP models. The ten-fold cross-validation we adopt when dividing the dataset using the bagging technique.

Stacking represents another powerful ensemble learning technique, involving the combination of several weak learners using meta-learners. These weak learners are trained independently, and their predictions are then employed as input for the meta-learner, which makes the final decision. Nunes [12] conducted a study utilizing stacking in a document classification task, showcasing its efficacy. We use a problem type classifier that allows the different solvers to play to their strengths.

Moreover, ensemble learning has shown promise in enhancing deep learning models in NLP. Kim [7] conducted a study where they employed an ensemble approach to improve the text classification performance of Convolutional Neural Networks (CNNs). The SC method we utilize in LLM solver is more like a self-ensemble, acting on a single language model.

2.2 Tree-Based MWP Solver

Early solvers in the field of MWP solving employed predefined patterns to map problems. To address this, a slot-filling mechanism was developed, enabling the mapping of problems into equation templates using slots [2–4]. Wang [17] introduced a sequence-to-sequence (Seq2Seq) approach for generating mathematical expressions. However, Huang [5] identified an issue with Seq2Seq models that predicted numbers in incorrect positions and generated incorrect values.

To address the problem of equation repetition, Wang [15] employed equation normalization techniques. Additionally, Xie and Sun [19] proposed a goal-driven tree structure (GTS) model, which greatly enhanced the performance of traditional Seq2Seq methods by generating expression trees. More recently, researchers have explored the utilization of pre-trained language models, such as BERT [6], in MWP solving. Peng [13] proposed an extension of BERT by incorporating numerical information into the input sequence, thereby enhancing the power of BERT in handling MWPs. In this paper, we adopt the sequence-to-tree approach with bert (**Bert2Tree**) to solve MWPs, leveraging its improved performance over traditional methods.

2.3 LLM Solver

In recent years, LLMs have showcased their remarkable capabilities in the field of NLP. Wei's [22] research explored the emerging capabilities of LLMs in solving MWPs through step-by-step reasoning, leveraging cues derived from the chain-of-thought (CoT) [18]. Without avoiding the greedy decoding strategy in the CoT, wang [16] proposed the SC method, which allowed multiple inference paths to reach the correct answer for complex reasoning tasks. In this work, we utilize ChatGLM-6B [20] as our LLM solver.

3 Research Methodology

Fig. 1 shows the overview of our method (Ensemble-MWP), which contains four main components: problem type classifier, Bert2Tree solver, LLM solver, and post-processing stage. Firstly, the problem type classifier assigns each MWP to either the tree-based solver or the LLM solver. Once the classification is determined, the respective solver is employed to process the MWP and generate a preliminary result. Lastly, the final result is obtained through a post-processing block. In the following, we will describe more details of each component.

3.1 Problem Type Classifier

In our proposed problem type classifier, we integrate the Bert2Tree solver and the LLM solver. The main objective of the classifier is to categorize the MWPs in the dataset into two categories. The first category comprises MWPs that can be effectively solved by the Bert2Tree solver. Consequently, these MWPs are directed to the Bert2Tree solver for further processing. The second category consists of MWPs that are beyond the capabilities of the Bert2Tree solver. For this category, we utilize the LLM solver to handle them.

The classification process is guided by specific heuristic rules to identify particular problem types. For instance, problems involving unit conversions (e.g., centimeters, decimeters, meters), law finding, and decimal point transformations are categorized as MWPs that the Bert2Tree solver is unable to solve. As a result, these specific problem types are directed to the LLM solver, which is better equipped to address them. By employing these heuristic rules, we effectively determine the appropriate solver for each MWP based on its characteristics.

3.2 Bert2Tree Solver

Model Structure: As illustrated in Fig. 2, the Bert2Tree model is employed to solve the MWP. Firstly, we input the question text into the Bert2Tree model. Secondly, the model encodes the question text and generates the corresponding equation tree. Thirdly, we calculate $8 \div 2 = 4$ according to the equation tree. Finally, the Bert2Tree model returns the answer of 4.0.

For the structure of Bert2Tree, we adopt BERT as our encoder, we represent the question $\boldsymbol{Q}$ as a sequence of $\boldsymbol{T}$ tokens: $\boldsymbol{Q} = [\boldsymbol{q_1}, \boldsymbol{q_2}, ..., \boldsymbol{q_T}]$ and the process of encoding is

$$[\boldsymbol{h_1^q}, \boldsymbol{h_2^q}, ..., \boldsymbol{h_T^q}] = \boldsymbol{BERT}\,([\boldsymbol{q_1}, \boldsymbol{q_2}, ..., \boldsymbol{q_T}]), \tag{1}$$

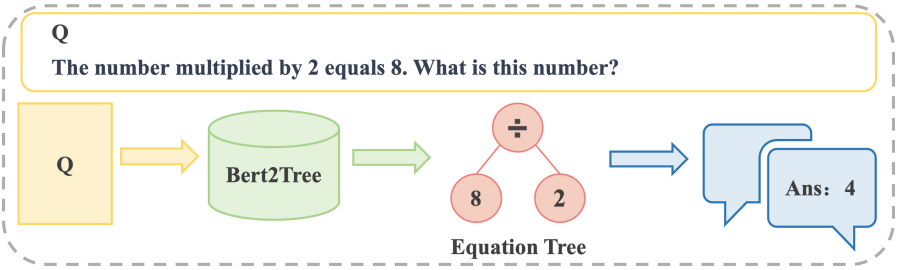

Fig. 2. The process of Bert2Tree solver. The MWP is fed into the **Bert2Tree model**, which performs a comprehensive analysis of the MWP text. Bert2Tree generates an **equation tree** that accurately captures the underlying mathematical structure of the problem. By extracting the mathematical expression from the tree, the model is able to compute the answer

where $\boldsymbol{h_i^q}$ represents the embedding of token $\boldsymbol{q_i}$ from the encoder. At last, the representation of question is $\boldsymbol{H^Q}$:

$$\boldsymbol{H^Q} = [\boldsymbol{h_1^q}, \boldsymbol{h_2^q}, ..., \boldsymbol{h_T^q}]. \tag{2}$$

Then, we use a TreeDecoder to generate equation tree $\boldsymbol{ET}$ according to $\boldsymbol{H^Q}$: $\boldsymbol{ET} = [\boldsymbol{et_1}, \boldsymbol{et_2}, ..., \boldsymbol{et_n}]$, where n is the length of the pre-order of equation tree, it can be written as:

$$[\boldsymbol{et_1}, \boldsymbol{et_2}, ..., \boldsymbol{et_n}] = \boldsymbol{TreeDecoder}\left(\boldsymbol{H^Q}\right). \tag{3}$$

Finally, calculate the equation tree can get the final $\boldsymbol{Answer}$:

$$\boldsymbol{Answer} = \boldsymbol{Calculate}\left(\boldsymbol{ET}\right). \tag{4}$$

Ten-Fold Cross-Validation: In this paper, we use a ten-fold cross-validation method to avoid overfitting and improve the generalization performance of the model. We break the dataset into 10 equal parts randomly: $\boldsymbol{D_0} - \boldsymbol{D_9}$. In the first model, $\boldsymbol{D_9}$ is used as the validation set, and the remaining 9 parts are used as the training set to train the model and make predictions on the validation set. The accuracy of the model on the validation set is calculated and recorded. For the next 9 models, a different copy of the data is used as the validation set each time, and the remaining 9 copies are used as the training set. Finally, we get the accuracy of the 10 sets of models to evaluate the performance of the models.

Voting Mechanism: In the process of improving the answer accuracy, we use the voting mechanism of ensemble learning to improve the probability of predicting the correct answer. When predicting the answer to the problem, there are two cases in our voting mechanism: (1) Different models have different predictions, so we first choose the one with the most occurrences among the models

as the final answer. (2) In cases where we have an equal number of voting results, we compare the sum of the accuracy of the validation sets of the models with the same prediction results, we set the accuracy of validation sets as the **confidence score** and select the one with the largest sum of confidence score as the final answer.

MWP-Bert and Data Augmentation: To further improve the ability of the Bert2tree solver, we use better encoder and data augmentation strategies. For the encoder, we use MWP-Bert [10], an MWP-specific pre-training language model. For data augmentation, we use Li's strategies [9]. They generate new MWPs by knowledge-guided entity replacement and logic-guided problem reorganization.

3.3 LLM Solver

In this paper, we utilize ChatGLM-6B as our LLM solver. To improve the performance of ChatGLM-6B, we use the Chain-of-Thought and Self-Consistency techniques.

Chain-of-Thought (CoT): The LLM solver relies primarily on the widely adopted CoT prompting [18], which has gained popularity in recent years. Few chain of thought demonstrations provided as exemplars in prompting can significantly improve the ability of large language models to perform complex reasoning. Specifically, we provide 8 MWPs in prompt and manually annotate detailed CoT for each MWP example. This enables the solver to acquire a comprehensive understanding of the problem-solving process and develop the capacity to apply logical thinking to mathematical challenges.

Self-Consistency (SC) Method: In the LLM solver, we leverage the SC [16] method as an integral component of our approach. It samples a diverse set of reasoning paths instead of only taking the greedy one and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Specifically, we generated 20 answers for each MWP. By incorporating the SC method into our LLM solver, we enhance the accuracy and robustness of the generated solutions, enabling more reliable and effective MWP solving.

3.4 Post-processing

Upon obtaining results from both models, an additional step is carried out to process the answers using uniform rules, leading to the derivation of the final results. This post-processing stage plays a crucial role in improving the accuracy rate by applying specific rules to refine the answers. For example, one common rule involves retaining only two digits after the decimal point, ensuring precision and consistency in the results. Additionally, certain rules may involve omitting trailing zeros, eliminating any unnecessary redundancy or ambiguity in the

final answers. These tailored rules provide a systematic approach to refine and standardize the answers, addressing any potential inconsistencies or inaccuracies generated by the individual models.

4 Experiments

4.1 Dataset Setting

We adopt Math23K [17] as our training dataset, which comprises 23,162 MWP examples. To ensemble Bert2Tree models, we use ten-fold cross-validation and obtain ten models. Each model's training dataset consists of 20,844 MWP examples, and the remaining 2,316 examples are included in the validation dataset. To evaluate our proposed Ensemble-MWP, we conduct experiments on a validation set of the NLPCC2023 Shared Task3 completion[1], which consists of 1,200 MWP examples. It is challenging to solve these MPWs because they have a low overlap of lexicon and templates with the Math23K dataset. We call these samples **Challenging Examples**, which make the models more difficult to generalize from the patterns and relationships seen in the training data.

4.2 Experimental Settings

To compare the model performance, we adopt the answers' accuracy as our evaluation metric, which is calculated by comparing the predicted answer with the correct answer. The higher the accuracy, the better the effect of the model or method used. For the baseline MPW solver, we adopt Bert2Tree [19] without voting mechanism and LLM solver [20] without SC as our baseline model in the experiments.

4.3 Experimental Results

Bert2Tree Solver: Through ten-fold cross-validation, we obtain the accuracy of each model, allowing us to compare the accuracy of different models. The results are shown in Table 2, where we can see that the accuracy ranges from 23.2% to 25.2%. Furthermore, we adopt MWP-Bert and Data augmentation on each Bert2Tree solver. In Table 3, we compared the accuracy of the model with and without the voting mechanism. We observed that when we used the voting mechanism, the accuracy improved from 24.1% to 26%. These results demonstrate that the voting mechanism is useful in solving MWPs correctly. When comparing Table 2 and 4, we see that the accuracy of the ten models are all improved when using MWP-BERT and data augmentation. It shows that using a domain-specific pre-trained language model like MWP-BERT and data augmentation can lead to better performance. As shown in Table 5, the accuracy was further improved by 2.8% when using the voting mechanism. It indicates that our voting mechanism is also efficient in stronger models.

[1] https://github.com/2003pro/CNMWP/tree/main/data.

Table 2. The performance of ten Bert2Tree solvers

Model (BERT)	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
Accuracy (%)	23.9	24.8	23.9	23.2	23.8	24.4	23.4	24.4	23.8	25.2

Table 3. Comparison of different methods. The accuracy of the baseline is the average of the ten models' accuracy in Table 2. The accuracy of the VoteMWP is the accuracy achieved by using the voting mechanism

Methods (BERT)	Baseline	**VoteMWP**
Accuracy (%)	24.1	**26.0**

Table 4. The performance of 10 Bert2tree solvers with MWP-Bert and data augmentation (DA)

Model (MWP-BERT+DA)	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
Accuracy (%)	25.8	25.8	26.1	26.6	25.4	24.5	26.1	24.3	25.6	25.5

Table 5. Comparison of different methods. The accuracy of the baseline is the average of the ten models' accuracy in Table 4. The accuracy of the VoteMWP is the accuracy achieved by using the voting mechanism

Methods (MWP-BERT+DA)	Baseline	**VoteMWP**
Accuracy (%)	25.6	**28.4**

LLM Solver: As we can see in Table 6, after incorporating the CoT prompting technique, the LLM solver initially achieves an accuracy rate of 5%. With the addition of the SC method, the accuracy rate of the LLM solver significantly improves to 9.17%. This enhancement demonstrates the effectiveness of integrating the SC method, as it directly contributes to the improved performance and reliability of the LLM solver in solving MWPs.

Problem Type Classifier: In our comparative analysis, we evaluate the performance of the Bert2Tree solver, the LLM solver, and the Ensemble-MWP solver. The results demonstrate that the integrated Ensemble-MWP solver achieves significantly higher accuracy compared to the individual solvers in Table 7.

By combining the strengths of multiple solvers and leveraging ensemble techniques, our integrated Ensemble-MWP solver offers improved capabilities in solving MWPs. The collaborative nature of the ensemble approach allows for the aggregation of insights and decision-making from multiple solvers, resulting in enhanced accuracy.

Table 6. Comparison of different methods in the LLM Solver. CoT prompting is used in the LLM solver, and we add the SC method later

Methods	CoT	**CoT + SC**
Accuracy (%)	5.00	**9.17**

Table 7. Comparison of different MWP Solvers. Three MWP Solvers: Bert2Tree Solver, LLM Solver, Ensemble-MWP

Solvers	Bert2Tree	LLM	**Ensemble-MWP**
Accuracy (%)	28.4	9.17	**33.1**

4.4 Case Study

In Fig. 3, we present a real case of Ensemble-MWP to illustrate the challenges faced when using a single solver. When both questions are inputted into a single solver, whether it is the Bert2Tree solver or the LLM solver, it is impossible to answer both questions correctly. However, by employing Ensemble-MWP, we utilize a problem type classifier that assigns each problem to the appropriate solver, resulting in accurate and reliable solutions for both questions. Through this ensemble approach, the final correct results are obtained, overcoming the limitations of using a single solver for multiple math word problems.

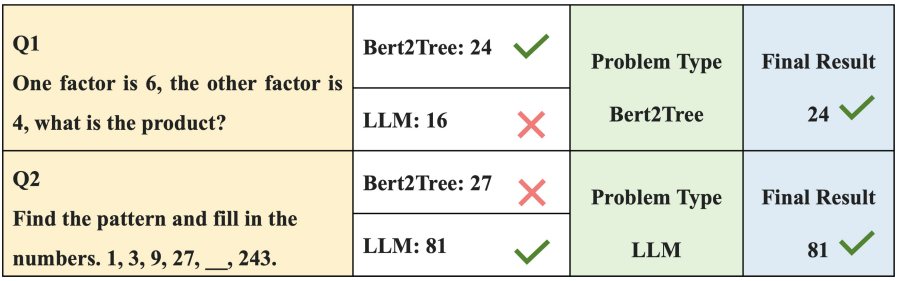

Fig. 3. Two cases solved by Ensemble-MWP

5 Conclusion and Future Work

In this paper, we propose an ensemble technique to enhance the capability of the MWP solver. By combining the strengths of the Bert2Tree solver and the LLM solver, we significantly improve the overall MWP-solving performance. Our approach capitalizes on the unique advantages offered by each solver, resulting in a novel and effective solution. Within the Bert2Tree solver, we introduce a ten-fold

cross-validation and voting mechanism to further enhance the model's robustness and reliability. Through multiple iterations of cross-validation, we rigorously evaluate the performance of the solver on different subsets of the data. The integration of the voting mechanism ensures robust decision-making by considering the collective insights of the model's predictions. These enhancements not only improve the accuracy of the Bert2Tree solver but also bolster its resilience to handle diverse MWPs effectively.

In the future, our goal is to develop an automatic classifier that can proficiently identify the appropriate solver for MWPs. This innovative approach aims to alleviate the reliance on predefined rules, consequently enhancing the robustness of the system. By leveraging machine learning techniques, the classifier will autonomously categorize MWPs, assigning them to the most suitable solver based on their unique characteristics.

Acknowledgments. This research was funded by National Natural Science Foundation of China (NSFC) no.62276258, Jiangsu Science and Technology Programme (Natural Science Foundation of Jiangsu Province) no. BE2020006-4, Xi'an Jiaotong-Liverpool University's Key Program Special Fund no. KSF-E-43.

References

1. Bobrow, D.G.: Natural language input for a computer problem solving system (1964). www.api.semanticscholar.org/CorpusID:56584838
2. Bobrow, D.G.: A question-answering system for high school algebra word problems. In: Proceedings of 1964 Fall Joint Computer Conference, Part I, pp. 591–614 (1964)
3. Dellarosa, D.: A computer simulation of children's arithmetic word-problem solving. Behav. Res. Methods Instr. Comput. **18**(2), 147–154 (1986)
4. Fletcher, C.R.: Understanding and solving arithmetic word problems: a computer simulation. Behav. Res. Methods Instr. Comput. **17**(5), 565–571 (1985)
5. Huang, D., Liu, J., Lin, C.Y., Yin, J.: Neural math word problem solver with reinforcement learning. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 213–223 (2018)
6. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, vol. 1, p. 2 (2019)
7. Kim, Y., Jernite, Y., Sontag, D., Rush, A.: Character-aware neural language models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
8. Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., Ang, S.D.: Parsing algebraic word problems into equations. Trans. Assoc. Comput. Linguistics **3**, 585–597 (2015)
9. Li, A., Xiao, Y., Liang, J., Chen, Y.: Semantic-based data augmentation for math word problems. In: Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part III, pp. 36–51. Springer (2022)
10. Liang, Z., Zhang, J., Wang, L., Qin, W., Lan, Y., Shao, J., Zhang, X.: Mwp-bert: numeracy-augmented pre-training for math word problem solving. arXiv preprint arXiv:2107.13435 (2021)

11. Lin, S., Kung, Y., Leu, F.: Predictive intelligence in harmful news identification by bert-based ensemble learning model with text sentiment analysis. Inf. Process. Manag. **59**(2), 102872 (2022)
12. Nunes, R.M., Domingues, M.A., Feltrim, V.D.: Improving multilabel text classification with stacking and recurrent neural networks. In: Silva, T.H., Dorini, L.B., Almeida, J.M., Marques-Neto, H.T. (eds.) WebMedia '22: Brazilian Symposium on Multimedia and Web, Curitiba, Brazil, November 7–11, 2022, pp. 117–122. ACM (2022)
13. Peng, S., Yuan, K., Gao, L., Tang, Z.: Mathbert: A pre-trained model for mathematical formula understanding. arXiv preprint arXiv:2105.00377 (2021)
14. Shi, S., Wang, Y., Lin, C.Y., Liu, X., Rui, Y.: Automatically solving number word problems by semantic parsing and reasoning. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1132–1142. Association for Computational Linguistics, Lisbon, Portugal, September 2015
15. Wang, L., Wang, Y., Cai, D., Zhang, D., Liu, X.: Translating a math word problem to an expression tree. arXiv preprint arXiv:1811.05632 (2018)
16. Wang, X., et al.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
17. Wang, Y., Liu, X., Shi, S.: Deep neural solver for math word problems. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 845–854 (2017)
18. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural. Inf. Process. Syst. **35**, 24824–24837 (2022)
19. Xie, Z., Sun, S.: A goal-driven tree-structured neural model for math word problems. In: Ijcai. pp. 5299–5305 (2019)
20. Zeng, A., et al.: Glm-130b: An open bilingual pre-trained model. arXiv preprint arXiv:2210.02414 (2022)
21. Zhou, Z., Ning, M., Wang, Q., Yao, J., Wang, W., Huang, X., Huang, K.: Learning by analogy: Diverse questions generation in math word problem. In: Findings of the Association for Computational Linguistics: ACL 2023, pp. 11091–11104. Association for Computational Linguistics, Toronto, Canada, July 2023. www.aclanthology.org/2023.findings-acl.705
22. Zoph, B., et al.: Emergent abilities of large language models. TMLR (2022)