Numerical Reasoning in NLP: Challenges, Innovations, and Strategies for Handling Mathematical Equivalency

Qianying Liu

August 2023

Abstract

Numerical reasoning plays a pivotal role in both the realm of natural language processing (NLP) research and real-life applications. Numerical reasoning, as an integral component of NLP research, assumes a critical role in the field of artificial intelligence. It is regarded as a crucial metric for assessing a model's capacity for reasoning and stands as a vital element in the pursuit of achieving Artificial General Intelligence (AGI). Leveraging numerical reasoning, researchers can develop algorithms and models that exhibit the ability to comprehend, generate, and analyze data from science, finance, news, and various other domains. This application of numerical reasoning facilitates advancements in domains such as language translation, sentiment analysis, and text summarization. In real life, the ability to understand and manipulate numbers is essential in various contexts, such as personal finance, data analysis, and problem-solving. Whether it's calculating budgets, interpreting statistical information, or making informed decisions based on quantitative data, numerical reasoning empowers individuals to navigate the complexities of modern life. The integration of numerical reasoning and NLP not only enhances everyday experiences but also propels the boundaries of artificial intelligence in comprehending and processing human language.

This thesis examines the significance of numerical reasoning and its research trajectory, followed by an introduction to relevant tasks and datasets. The primary focus of this research lies in highlighting a key characteristic of these tasks, which involves the presence of answers with **mathematical equivalency**. Consequently, such tasks present numerous challenges, we propose various methods to address these challenges.

The initial section of this thesis provides a comprehensive overview of numer-

ical reasoning, emphasizing its importance across various domains. Additionally, the historical background of research in this field is explored, shedding light on the progress made thus far and identifying the gaps that remain.

Subsequently, the thesis delves into the specifics of the tasks and datasets associated with numerical reasoning. By examining these elements, the research establishes a foundation for comprehending the intricacies and complexities inherent in this domain. Building upon previous research, this thesis focuses on the identification and analysis of a crucial aspect of numerical reasoning tasks, that the answer equation of one problem could have multiple equivalent expressions, namely mathematical equivalency. This aspect introduces a range of challenges that need to be addressed for effective problem-solving.

The **Costly Expert Annotation Dilemma** is explored, highlighting the need for efficient and scalable methods to annotate training data, considering the inherent complexity of numerical reasoning. We propose a novel data augmentation method that reverses the mathematical logic of math word problems to produce new high-quality math problems and introduce new knowledge points that can benefit learning the mathematical reasoning logic. We also propose three pretraining tasks that operate at both the whole program and sub-program level, which guides the model to focus on useful variables and encourages the model to identify key evidence.

Furthermore, the Challenge of **Mismatch of Supervised Training Objective** is addressed, as it poses a significant obstacle to training models effectively. The thesis explores methods to reduce the training noise caused by the mismatch and improve the robustness of the models, enabling them to handle real-world scenarios and noisy data. We propose a Textual Enhanced Contrastive Learning framework, which enforces the models to distinguish semantically similar examples while holding different mathematical logic. We adopt a self-supervised manner strategy to enrich examples with subtle textual variance by textual reordering or problem re-construction. We then retrieve the hardest to differentiate samples from both equation and textual perspectives and guide the model to learn their representations. We also propose a controlled equation generation solver by leveraging a set of control codes to guide the model to consider certain reasoning logic

and decode the corresponding equations expressions transformed from the human reference.

Lastly, the thesis investigates the **Enormous Search Space and False-Matching in Weak Supervision** challenge, which involves navigating through a vast array of possible answers, some of which may appear correct but lack mathematical equivalency. Strategies and techniques are proposed to mitigate the impact of this challenge and enhance the accuracy of numerical reasoning systems. We propose a novel search algorithm with the combinatorial strategy, which can compress the search space by excluding mathematically equivalent equations. The compression allows the searching algorithm to enumerate all possible equations and obtain high-quality data. We investigate the noise in the pseudo labels that hold wrong mathematical logic, which we refer to as the *false-matching* problem, and propose a ranking model to denoise the pseudo labels.

By examining and understanding the challenges associated with numerical reasoning tasks, this thesis seeks to contribute to the advancement of this field. Through these five proposals, we have effectively addressed the issue of mathematical equivalency while significantly enhancing system performance. The proposed solutions and strategies aim to enhance the accuracy and reliability of numerical reasoning systems, facilitating their practical application across a range of domains.

Acknowledgments

I am profoundly grateful as I reflect upon my seven-year journey delving into NLP research. Amidst a series of fortuitous events and unforeseen challenges, I found myself grappling with existential nihilism after the onset of the Covid pandemic. However, my relentless pursuit of knowledge and the ensuing anxiety it brought acted as guiding lights, helping me traverse the abyss of self-isolation and eventually, albeit with difficulty, complete this PhD study.

First and foremost, I extend my heartfelt gratitude to my advisor, Sadao Kurohashi, who has never faltered in providing unwavering support and ample resources, enabling me to stumble and struggle through every predicament. I must also acknowledge my co-advisor, Fei Cheng, whose thorough involvement in my research not only supplied invaluable insights and sparks of inspiration but also urged my nihilistic self to embrace the realm of reality.

I wish to express my appreciation to other staff members in the laboratory: Chenhui Chu, Yugo MURAWAKI, for their valuable advice and guidance on my research, and to my fellow students Zhuoyuan Mao, Zhen Wan, Zhengdong Yang, Haiyue Song, for their cherished academic collaborations and camaraderie throughout both academic and personal spheres.

I am indebted to my dear friends Haoran Zhang, Wanqing Zhu, Liang Shan, Qiyao Wang, Yufan Yang, Zeyu Liu, Ruoye Lv, Shuying Zhang, who accompanied me through this journey, offering emotional support and assistance in various aspects.

I am grateful to my academic collaborators Yibin Shen, Wenyv Guan, Wenjie Zhong, Zhuo Gong, and others, with whom I have worked on various projects, enriching my experience in numerous ways.

I extend my heartfelt appreciation to my internship mentors, Tianyu Zhao, Sheng Li, and Chenchen Ding, for their tremendous support during my internship.

I am grateful to my family, especially my great-grandparents, for their unwavering care and concern.

Lastly, I would like to thank Dongsheng Yang for being by my side throughout my PhD journey.

I am truly honored to have found both something to love, the realm of research, and someone to cherish during this period of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the realm of Natural Language Processing (NLP) and Artificial Intelligence (AI), the ability to reason with numerical information has emerged as a crucial aspect, enabling machines to navigate and comprehend the complexities of human language. Numerical reasoning encompasses the capacity to interpret, analyze, and draw meaningful insights from quantitative data embedded within textual or tabular content. By integrating numerical reasoning capabilities into NLP and AI systems, we could unlock a plethora of possibilities, ranging from advanced question-answering systems to automated decision-making algorithms.

The rise of big data and the proliferation of textual and multimedia information have generated unprecedented challenges in the field of NLP. While traditional NLP approaches have successfully addressed syntactic and semantic aspects of language, incorporating numerical reasoning opens new avenues for understanding complex relationships and phenomena concealed within massive datasets. By augmenting machines with the ability to comprehend and manipulate numerical information, we enable them to navigate intricate domains such as finance, healthcare, and scientific research, where quantitative analysis plays a pivotal role. In Figure 1.1, we provide an example where an assistant AI with can automatically understand and calculate a client's monthly expenses on different projects such as food through numerical reasoning.

Furthermore, the integration of numerical reasoning within NLP and AI systems offers immense potential for real-world applications. From analyzing finan-

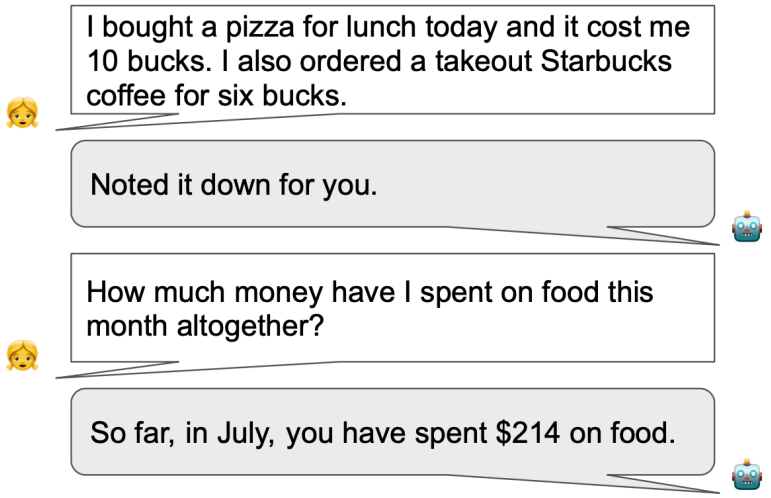


Figure 1.1: Example of numerical reasoning in real-life application.

cial reports and predicting market trends to understanding medical data and assisting in evidence-based diagnoses, numerical reasoning empowers machines to contribute meaningfully across various sectors. With the ability to process and reason about numbers, machines can provide insights, generate accurate predictions, and assist in decision-making processes, thereby revolutionizing industries and enhancing human-machine collaborations.

This thesis aims to delve into the realm of numerical reasoning within NLP and AI, shedding light on its fundamental concepts, methodologies, and its pivotal role in modern information processing systems. We investigate existing methodologies, from rule-based systems to and machine learning algorithms, and recent large language models development, which enable machines to extract numerical information and reason with it effectively. We point out the key challenge of numerical reasoning task, namely mathematical equivalency, and propose five methods addressing three major challenges caused by it. By comprehending the nuances of numerical reasoning, we can design more robust and intelligent systems capable of unlocking the potential hidden within vast volumes of textual and numerical data.

Ultimately, this research seeks to highlight the critical importance of numerical

reasoning within NLP and AI. By unraveling the significance of this multidisciplinary field, we aim to pave the way for the development of more advanced and capable information processing systems. As we embark on this journey, we anticipate that the insights gained from this study will contribute to the ongoing evolution of NLP and AI, enhancing their capacity to comprehend, reason, and derive valuable insights from the ever-expanding realm of human language and numerical data.

In this chapter, we will first delve into the historical development of numerical reasoning, tracing its evolution within the field of NLP and AI. By examining the progress made thus far and presenting novel methodologies, we aim to contribute to the advancement of numerical reasoning in NLP and AI, ultimately fostering the development of more sophisticated and effective information processing systems.

1.1 History of Numerical Reasoning

1.1.1 Rule-based systems

The section presents an overview of various early rule-based systems developed for the purpose of reading and comprehending algebraic and arithmetic word problems, with the aim of providing solutions and answers in natural language.

The pioneering system discussed is the **STUDENT** program [7]. This program represents a significant milestone as it was the first to read and understand algebraic problems expressed in restricted English language and respond in English as well. By utilizing a relational model, **STUDENT** effectively stores information and transforms complex sentences into simpler equations, ultimately solving the given problems using established methods.

The **DEDUCOM** [98] focuses on storing LISP expressions of data statements and employing conditional statements to infer answers. While **DEDUCOM** showcased promising capabilities, it was found to be relatively slow in its responses and faced challenges in logical deduction, particularly within predicate calculus. However, it demonstrated improved performance when provided with more relevant facts arranged in an appropriate order.

WORDPRO [30] tackles the understanding and resolution of arithmetic word

problems. It employs a set of propositions to represent the semantic meaning of the problem text, employing the concept of set schemas for problem modeling. **WORDPRO** relies on a sequential application of rules, arithmetic strategies, and problem-solving procedures, leveraging the content of its short-term memory to guide the problem-solving process.

Other rule-based systems include **CHIPS** [8], **ARITHPRO** [22], and **ROBUST** [3]. **CHIPS** and **ARITHPRO** focus on solving one-step arithmetic word problems, predominantly involving addition and subtraction operations. While these models categorize problems into compare, combine, and change scenarios, they exhibit limitations concerning the choice of verbs and the sentence order within the problem text. On the other hand, **ROBUST** excels in understanding free-format multi-step arithmetic word problems, even when extraneous information is present. ROBUST utilizes an expanded concept of "change formulae" and employs comprehensive schema categories to describe different types of changes and solve complex problem scenarios.

Finally, several computer-aided instruction (CAI) systems are designed to assist students in learning how to solve algebraic and arithmetic word problems. These systems, including **PAT** [50], **WORDMATH** [69], **DISCOVER** [99], **WPS Tutor** [107], and **MathCAL** [12], offer predefined strategies and instruction based on model problems stored in their databases. However, they lack the capability to comprehend and solve novel, unseen problems presented by the learners.

In summary, the various systems reviewed in this section have contributed to the development of computational methods for tackling algebraic and arithmetic word problems. From early systems like **STUDENT** and **DEDUCOM** to more recent advancements like **WORDPRO** and **ROBUST**, each system has employed distinct techniques to address the challenges associated with problem understanding and solving. While these advancements have paved the way for further research and innovation, however, these rule-based systems suffer from poor generalization which limits the real-life applications.

1.1.2 Statistic-based Methods

The section discusses statistic-based methods for solving arithmetic word problems using traditional machine learning models. These methods employ logic inference procedures to identify entities, quantities, and operators from the problem text and generate numeric answers. Roy et al. Roy et al. [93] proposes a quantity entailment scheme which utilizes classifiers to detect properties of the word problem, such as quantity pairs, operators, and operand order. By inferring expressions, the numeric answer can be straightforwardly calculated for simple math problems.

To tackle multi-step arithmetic expressions, statistic-based methods require advanced logic templates, which involve additional overhead for text problem annotation. For instance, **ARIS** [43] introduces a logic template called "state" that defines entities, containers, attributes, quantities, and relations. **ARIS** splits the problem text into fragments and tracks state updates based on verb categorization. Sundaram et al. [100] predefines a corpus of logic representations scheme and matches the sentences in the text problem to trigger update operations.

Mitra et al. [76] propose a new logic template formula to solve problems with addition and subtraction. They define formulas for part-whole, change, and comparison scenarios, allowing the conversion of text problems into algebraic equations. Liang et al. [60] converts the problem text into logic form representations using predefined mapping rules. Logic inference is then performed on these derived statements to obtain the answer.

Despite their usefulness, statistic-based methods have drawbacks. They require significant annotation effort, hindering their scalability for large-scale datasets. Moreover, these methods heavily rely on predefined templates, making them rigid and difficult to extend for supporting other operators or handling diverse datasets. As a result, tree-based solutions have gained prominence as mainstream approaches for solving arithmetic word problems.

Several tree-based approaches have been developed to solve arithmetic word problems by representing arithmetic expressions as binary tree structures. These approaches aim to construct equivalent tree structures in a bottom-up manner without the need for additional annotations [51, 90, 91]. The tree construction

process involves extracting quantities from the text, enumerating syntactically valid candidate trees, and selecting the best matching tree using a scoring function. These approaches utilize local classifiers to determine the likelihood of operators being selected as internal nodes and incorporate global scoring functions to evaluate the overall tree likelihood.

Roy et al. [90] introduced the concept of expression trees, reducing the search space by training a binary classifier to identify relevant quantities. The tree construction procedure maps to prediction problems determining lowest common ancestor operations. **ALGES** [51] explores all possible equation trees and uses Integer Linear Programming to enforce constraints, resulting in higher computational costs.

1.1.3 Deep Learning Model based Methods

Deep learning (DL) has demonstrated significant success in various smart applications. Several DL-based solvers have been developed for math word problem solving, leveraging effective feature representations learned in a data-driven manner without human intervention.

Deep Neural Solver (DNS) [104] is serves as a pioneering work in this domain which directly predicts the equation solution as a sequence decoding task, given the problem text input.

Graph modeling has gained popularity in math word problem solving [65, 102, 109, 115]. Approaches like modeling the input problem as a graph or using graph neural networks for both encoder and decoder have been explored. These graph-based models capture complex semantic relationships, enable the representation of linguistic and mathematical interactions, and show promising performance. **Unit-Dep** [91] introduces a Unit Dependency Graph to enhance the scoring function, considering associations between quantities and rates.

Contrastive learning approaches, leveraging Siamese networks and transformers, have been proposed to overcome the challenge of different mathematical structures in linguistically similar word problems [59, 96]. By designing semantically informed intermediate representations, these models aim to capture similarity based on both language and mathematical concepts.

Knowledge distillation and post-pretraining techniques, inspired by large pre-trained models in NLP, have been applied to math word problem solving [18, 24, 61]. By fine-tuning large generic networks on smaller datasets, task-specific models are distilled, focusing the learning of the generic model onto a more focused one. This approach shows promise in adding semantic information and improving performance, especially with limited data points.

These DL-based approaches showcase the application of deep learning techniques in math word problem solving, demonstrating their potential to handle complex problem structures and achieve accurate results.

In light of these observations, it could be contended that numerical reasoning within the era of deep learning extends beyond mere search and pattern memorization. Firstly, even when presented with a large number of variables, the potential solution space can reach billions, yet models still demonstrate a degree of generalization ability. This suggests that to some extent, they are engaging in reasoning processes. Moreover, the most developed deep learning based methods are capable of detecting subtle textual modifications, encompassing cases where only a couple of words are changed yet fundamentally altering the mathematical semantics, which results in a complete different equation solution. These advanced models demonstrate a notable level of robustness, underscoring their ability for nuanced reasoning rather than mere memorization of patterns.

1.1.4 Large Language Model based Methods

The rapid advancement of large language models has greatly propelled the field of natural language processing. These models have demonstrated remarkable proficiency in tasks such as text generation, translation, and sentiment analysis, among others. However, despite their impressive linguistic capabilities, their ability to perform mathematical reasoning remains limited, as the most advanced GPT-4 model cannot beat small size task specific models on numerical reasoning datasets such as DROP. Mathematical reasoning involves understanding and manipulating mathematical concepts, solving complex equations, and drawing logical inferences from numerical data. While large language models can handle basic arithmetic and simple mathematical operations, they often struggle with more

intricate mathematical reasoning tasks that require deep understanding of mathematical principles and techniques. This limitation poses a significant challenge in leveraging these models for applications that heavily rely on mathematical reasoning, such as automated theorem proving, advanced data analysis, and mathematical problem-solving. Therefore, further research and development efforts are needed to enhance the mathematical reasoning capabilities of large language models, bridging the gap between natural language processing and mathematical reasoning domains.

Few-shot in-context learning, also known as few-shot prompting, allows a model to process a few examples before attempting a task [9]. This technique gained popularity after the introduction of GPT-3 and is considered an emergent property of large language models when scaled properly.

Chain-of-thought prompting demonstrates how reasoning abilities naturally emerge in large language models. It involves providing a few demonstrations of chain-of-thought as exemplars in prompting. [62, 105, 118]

It is crucial to recognize that numerical reasoning is a complex and multi-faceted process, and fully addressing its challenges in large language models requires ongoing research and exploration of novel techniques. While these methods contribute to progress in the field, there is still a need for further advancements to achieve comprehensive and robust mathematical reasoning capabilities in natural language processing systems.

1.2 The Central Challenge in Numerical Reasoning: Mathematical Equivalency

The history of research in numerical reasoning has witnessed significant advancements across various methodological approaches. Despite the evolution of methodology, a key challenge of the task that sets the task apart from other multi-hop question answering tasks due to its inherent characteristic of having multiple mathematically equivalent possible solutions.

We show a multi-hop QA example of a popular benchmark HoppotQA in Figure 1.2, to find the answer *'Sancramento Kings'*, the model needs to first find

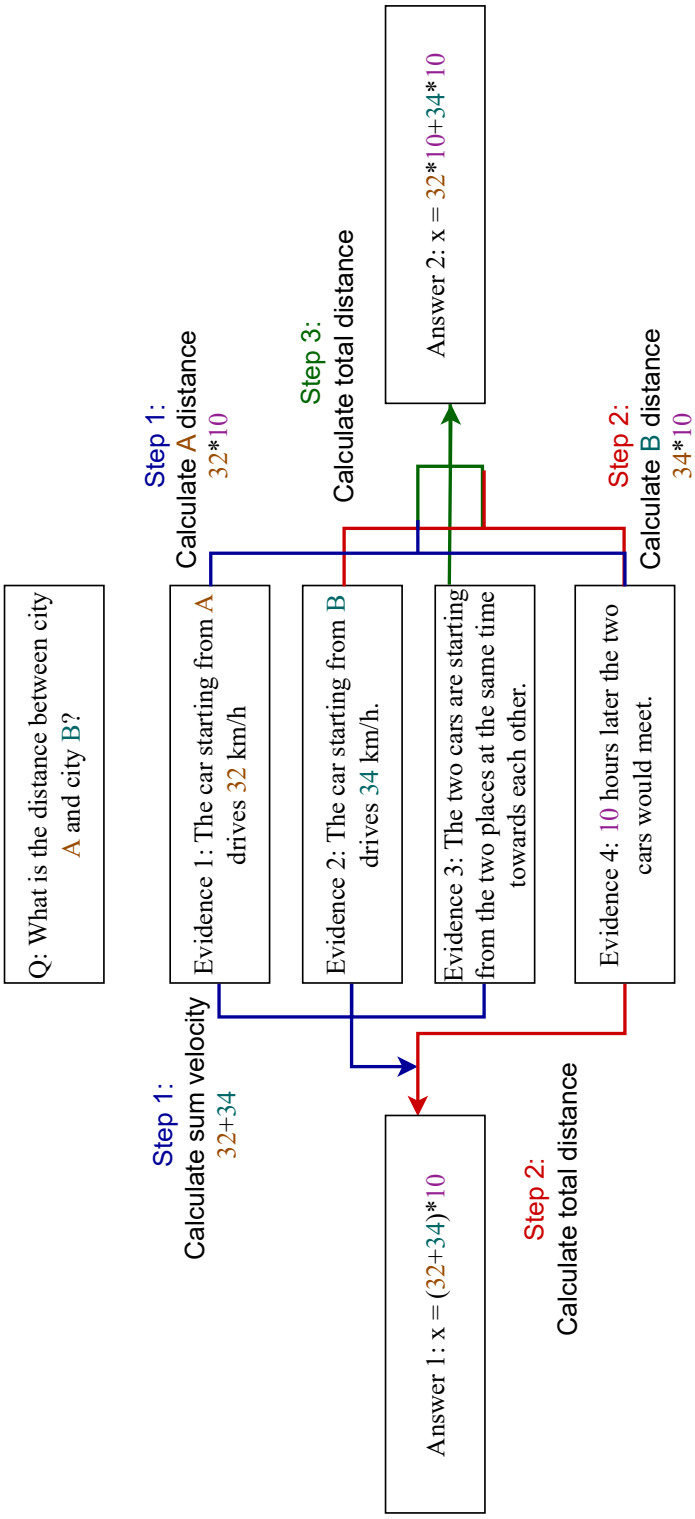


Table 1.1: The multiple possible reasoning path of numerical reasoning.

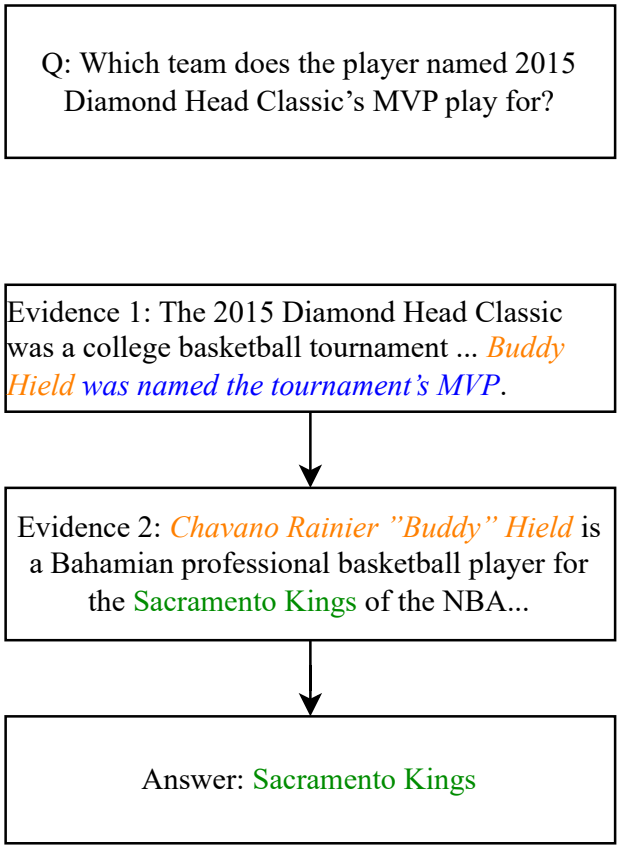


Figure 1.2: The determined reasoning path of multi-hop QA.

which player was named 2015 Diamond Head Classic's MVP, and then find which team Buddy Hield is playing for. The reasoning path for solving this question is determined.

On the other hand, numerical reasoning poses a unique challenge. Mathematical equivalency plays a crucial role in this context, as it allows for the existence of multiple valid solutions that satisfy the given constraints. We show two plausible reasoning paths of one math word problem in Figure 1.1. To calculate the total distance between city A and B, the reasoning path on the right side first calculates the individual distance that the two cars drive and then sums them up; the reasoning path on the left side first calculates the total velocity of the two cars and then calculate the total distance. These two reasoning paths are mathe-

matically equivalent but the expression of the solution equation is different. Such characteristics give rise to unique challenges in the task of numerical reasoning.

Various studies had pointed out this issue could harm model training and addressed it with normalization, regularization, and weak supervision. Wang et al. [102] normalized solution equation annotation so that various types of equivalent expressions are normalized to one fixed form. Contrastive learning [59] and adversarial training [95] based on mathematical equivalency were applied as regularization terms to guide the model beyond token level supervision. Some studies have suggested that utilizing weak supervision solely based on answer numbers can circumvent mathematical equivalency. However, existing methods still require searching for potential arithmetic expressions and cannot completely bypass this challenge [42].

In conclusion, the history of research in numerical reasoning has seen the evolution of rule-based, statistic-based, tree-based, deep learning-based, and large language model-based methods. While LLMs have demonstrated tremendous success across various NLP tasks, their performance in numerical reasoning lags behind DL-based methods. One of the most crucial research challenges lies in addressing the multitude of difficulties posed by mathematical equivalency. While numerous studies have been conducted to tackle this issue, they have proven insufficient. In the subsequent sections, we will provide a more detailed analysis of the challenges posed by mathematical equivalency and propose our own method to address them.

Chapter 2

Preliminary Background

In this section, we first introduce the specific task forms and relevant datasets of numerical reasoning in NLP. As shown in Figure 2.1, we then highlight the common characteristics of these tasks, emphasizing how their answers and annotations are influenced by mathematical equivalency, which sets them apart from general multi-hop reasoning and presents unique challenges. Within this thesis, we address three key challenges that have emerged in this domain, namely the Costly Expert Annotation Dilemma, Mismatch of Super, and Enormous Search Space and False-Matching. For each challenge, we propose corresponding solutions to tackle them effectively.

2.1 Tasks and Datasets of Numerical Reasoning

2.1.1 Solving Math Word Problems

Solving math word problems (MWP) is the task of mapping a natural language description of a mathematical problem into its solution. These mathematical problems were originally designed as application exercises for human students studying mathematics. As such, they were artificially created to provide practical contexts for applying mathematical concepts and skills. These problems are often used to test the numerical reasoning abilities of models. Researchers employ these problems as benchmarks to assess the performance and capabilities of NLP systems. By presenting these problems to the models, researchers can evaluate

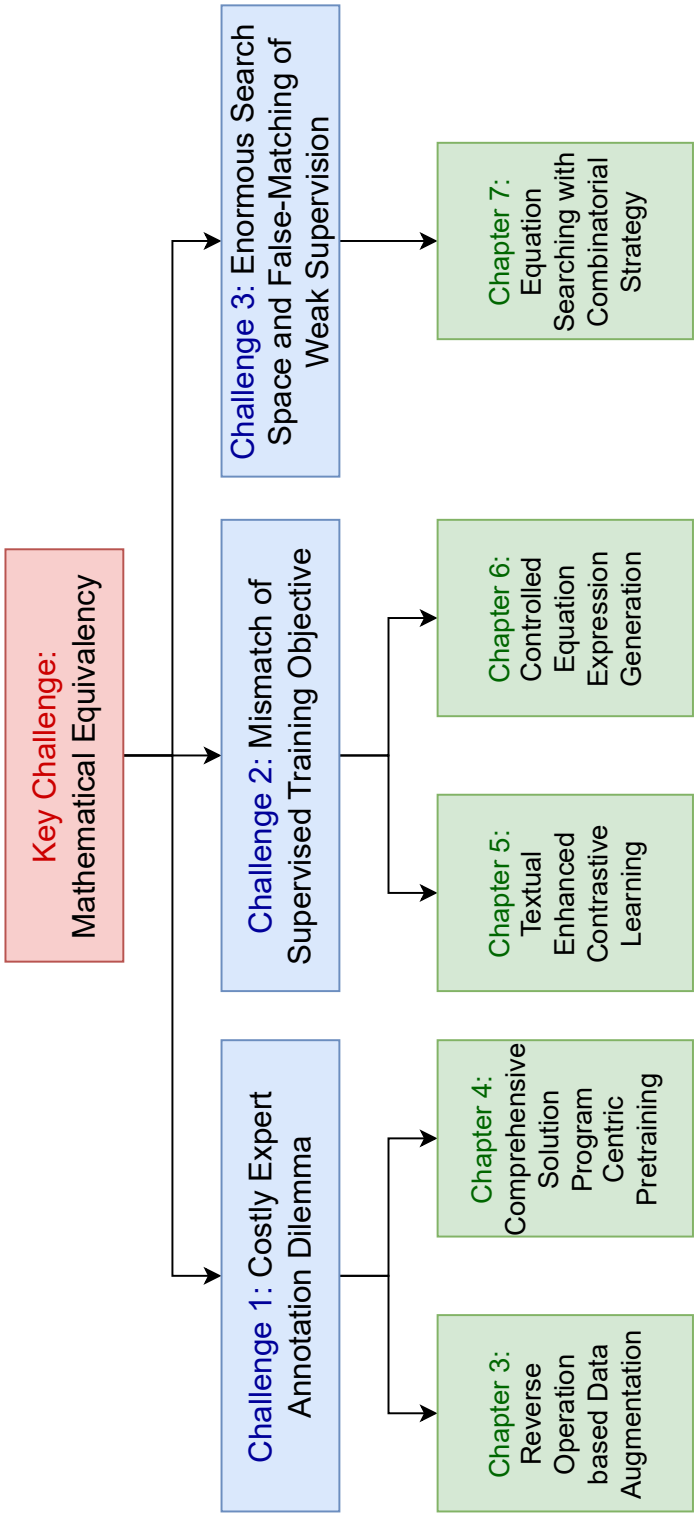


Figure 2.1: The main idea of this thesis.

Problem	The distance between the two places A and B is 660 km, the car starting from A drives 32 km/h, and the car starting from B drives 34 km/h. The two cars are starting from the two places at the same time in inverse direction. How many hours later would the two cars meet?
Equation	x = 660 / (32 + 34)
Answer	10

Figure 2.2: One example of MWP. Problem refers to the natural language descriptions. Equation refers to the formal math equation. Answer refers to the final quantity solution.

their ability to comprehend, analyze, and solve mathematical challenges.

As shown in Figure 2.2, under the supervised training setting, the model takes in the natural language description of the math word problem and predicts an equation solution sequence. Then the final answer is obtained by calculating the value of the equation solution. Under the weakly supervised training setting, only the answer value is given as gold annotation. The system extracts candidate solution equations via searching or sampling strategies.

Pioneer studies include ALG514 [56], a classical dataset comprising 514 word problems, as a foundation for addressing word problem challenges on small datasets. Hosseini et al. [43] introduced AddSub, focusing on simple addition and subtraction problems. Roy et al. [93] and Roy and Roth [90] proposed SingleOp and MultiArith, respectively, enabling control over the number of operators employed. Unique in its incorporation of long sentence structures for elementary-level school problems, SingleEq was introduced by Koncel-Kedziorski et al. [51]. AllArith, as presented by Roy and Roth [91], serves as a subset combining AddSub, SingleEq, and SingleOp. All datasets provide annotations for equations and answers.

With the advancement of deep learning techniques, researchers have been constructing larger datasets to train and evaluate models. The aim is to enhance the performance and generalization capability of these models. These larger-scale datasets not only provide richer samples and diverse features but also assist models in better understanding and capturing complex patterns and relationships within the data. MAWPS [52] is a curated dataset that includes previously proposed datasets, and AsDIV-A [73] is a single-equation subset of MAWPS for diagnostic analysis. Dolphin18k [44] is an early proprietary dataset evaluated primarily with statistical solvers. AQuA-RAT [63] introduced a large crowd-sourced dataset, offering explanations and rationales for word problems across various domains, distinguishing itself from earlier datasets in terms of size and annotation scope. Amini et al. [1] critically analyzed AQuA-RAT and created MathQA, a core subset annotated with a predicate list to expand its utility. It should be noted that MathQA is a subset of AQuA-RAT. GSM8k [21] is a recent single-equation dataset, representing a large-scale version of AsDIV-A [73]. Math23K is a popular Chinese dataset for single equation math word problem solving.

With deep learning approaches reaching performance levels comparable to human capabilities, researchers have noted that these models often focus on learning shallow patterns rather than engaging in genuine numerical reasoning. Consequently, multiple challenge datasets and adversarial datasets have been proposed to address this limitation and evaluate the models' ability to perform robust numerical reasoning tasks. SVAMP [81] and adv-AsDIV [55] are new datasets that build on AsDIV-A, challenging solvers to capture nuances in textual descriptions.

In this paper, we primarily focus on exploring datasets that center around 2-3 operators and primarily cater to the elementary level, based on the existing developments in the field and available models. Recently, more challenging datasets have been proposed, which extend beyond the scope of this study. We acknowledge that the datasets used in this research mainly revolve around basic arithmetic operations, typically suitable for elementary school students. However, it is important to note that in the future work section, we discuss the potential transferability and applicability of our proposed methods to address the emerging and more intricate datasets. We recognize the importance of addressing these

more complex datasets and believe that our approach's adaptability can pave the way for further advancements in the field.

2.1.2 Numerical Question Answering

In addition to math word problems designed specifically to assess students' mathematical abilities, there exists a broader category of question answering that utilizes numerical reasoning skills. DROP [25] is a reading comprehension dataset that requires the model to entail understanding a given passage and performing numerical calculations to arrive at a specific numerical answer. EQUATE [87] dataset considers quantitative natural language inference (NLI) questions. These questions involve performing simple arithmetic calculations to accurately classify the relationship between a given premise and hypothesis. Various studies investigate numerical reasoning with external commonsense knowledge, such as science formulas, chemistry, temporal commonsense knowledge, geometry and so on [75, 86].

Another research direction explores another common scenario in numerical reasoning, which involves tabular data. Tabular data plays a significant role in various domains, including finance, healthcare, and scientific research. This research line focuses on tasks that require understanding and reasoning with numerical information presented in tables, such as performing calculations, making comparisons, and extracting insights. Numerous studies have delved into question answering different types of tables, including database (DB) tables, flat web tables, and hierarchical tables [16, 19, 49]. These investigations aim to address the challenges associated with extracting relevant information from structured data sources and providing accurate answers to queries. Moreover, there has been a notable focus on domain-specific question answering tasks involving hybrid text-and-table data, particularly in financial domains [17, 116, 120]. This research direction has gained popularity due to the wide range of applications in areas such as financial analysis.

We show an example of numerical question answering in Figure 2.3. Given a relatively long passage with hybrid table-and-text information, the model extracts useful evidence corresponding to the question and then predicts the solution equation. The final answer value is obtained by calculating the equation. The

annotation of such datasets is usually annotated in semantic parsing program style.

2.2 Challenges and Proposed Methods

Numerical reasoning sets itself apart from other multi-hop question answering tasks due to its inherent characteristic of having multiple mathematically equivalent possible solutions. This characteristic of numerical reasoning directly gives rise to several challenges.

2.2.1 Costly Expert Annotation Dilemma

Mathematical equivalency poses challenges for crowd-sourced workers to achieve high annotation consistency, resulting in suboptimal quality. Therefore, in general, numerical reasoning datasets tend to rely on expert annotations to ensure higher quality and consistency, which significantly increases the cost of annotation compared to other NLP tasks. Therefore, it is crucial to leverage limited annotated samples effectively and develop strategies to address these challenges in order to mitigate the high cost and improve the quality of numerical reasoning models.

To address the aforementioned problem, we propose two innovative methods. Firstly, we introduce a data augmentation technique that leverages the concept of mathematical equivalency in Chapter 3. By reversing the mathematical logic of math word problems, we generate new high-quality math problems that introduce novel knowledge points. This augmentation process enriches the training data, enabling the model to learn the underlying mathematical reasoning more effectively.

Secondly, we propose a pretraining-based method to mitigate the impact of noise and irrelevant variables in the model input in Chapter 4. We observe that coarse-grained supervision of the whole solution program impedes the model's ability to learn the underlying numerical reasoning process. To overcome this challenge, we introduce three pretraining tasks operating at both the whole program and sub-program levels. These tasks include Variable Integrity Ranking,

Passage:
... We acquired the following apartment communities during the year ended December 31, 2017:

Community	Market	Units	Closing Date
Charlotte at Midtown	Nashville, TN	**279**	March 16, 2017
Acklen West End	Nashville, TN	**320**	December 28, 2017

... During the year ended December 31, 2017, we disposed of five multifamily properties totaling **1760** units and 4 land parcels totaling approximately 23 acres. ...
.*(abbreviate ~10 sentences)*

Question: During the year ended December 31, 2017, what was the ratio of the units disposed to the units acquired?

Equation: **1760** / (**279** + **320**)

Program:
divide(add(**279** + **320**), **1760**)

Answer: 2.938

Figure 2.3: An example of FinQA dataset.

which guides the model to focus on useful variables; Variable Operator Prediction, which decomposes the supervision into fine-grained single operator prediction; and Variable Keyphrase Masking, which encourages the model to identify key evidence from which sub-programs are derived.

Overall, our research addresses the challenge of costly annotation in numerical reasoning tasks by proposing innovative methods such as data augmentation and pretraining tasks. These approaches aim to enhance the training data quality and improve the model's ability to reason effectively in mathematical contexts.

2.2.2 Mismatch of Supervised Training Objective

During the supervised training process, mathematical equivalency causes a mismatch of supervised training objective and task target. The model may predict one possible mathematical solution, while the annotation provides another equivalent solution. This mismatch between model predictions and annotations can hinder the learning process and make it harder for the model to converge effectively.

To tackle the interference caused by equivalent equations, we propose two innovative methods. Firstly, we introduce a Contrastive Learning framework enhanced with textual information in Chapter 5. Solving math word problems requires a comprehensive understanding of contextual natural language information and the analysis of quantity relations. However, existing models often rely on shallow heuristics and can be easily misled by small textual perturbations. To overcome this, our Textual Enhanced Contrastive Learning framework enforces the model to distinguish semantically similar examples that hold different mathematical logic. Through a self-supervised approach, we introduce subtle textual variance by reordering or reconstructing problem statements. We then select the most challenging samples, both from an equation and textual perspective, to guide the model in learning their representations.

Secondly, we propose a method that leverages prompts to control equation generation in math word problem solving in Chapter 6. Human students employ diverse reasoning logics, which can lead to different potential equation solutions. In contrast, conventional automatic solvers based on sequence-to-sequence models

aim to decode a fixed solution equation supervised by human annotation. To address this limitation, our approach involves leveraging a set of control codes to guide the model's consideration of specific reasoning logics. These control codes facilitate the decoding of equation expressions that are transformed from human references, enabling the model to explore different reasoning pathways.

Overall, our research addresses the interference caused by equivalent equations during training in numerical reasoning tasks. Through the adoption of a Textual Enhanced Contrastive Learning framework and the utilization of prompts to control equation generation, we aim to improve the model's ability to handle the complexity of math word problems and enhance its reasoning capabilities in the presence of equivalent equations.

In this thesis, it is essential to acknowledge the collaborative efforts that have contributed significantly to the research presented herein. I wish to declare that Chapter 5 and Chapter 6 of this work are based on co-first author papers I have had the privilege of co-authoring alongside Yibin Shen. The collaboration with Yibin Shen has been instrumental in producing the valuable insights and findings within these chapters. I have also explicitly outlined my individual contributions to each respective chapter. Moreover, I have obtained the full and informed agreement from Yibin Shen to utilize the co-first author papers in the composition of this thesis.

2.2.3 Enormous Search Space and False-Matching of Weak Supervision

Despite the absence of the problem of equivalent expressions annotation, weakly supervised training still faces challenges arising from mathematical equivalency. This mathematical equivalency not only affects data annotated in the form of equations, but even when using a deterministic answer value as the annotation instead of an equation or rationale with multiple equivalent forms, the search space for preferred equations remains enormous due to mathematical equivalency. Moreover, numerical coincidences can still lead to mathematical logic errors, resulting in false-matching samples where the final answer is correct but the generated equations are incorrect. This introduces additional noise during the training

process, making it more challenging to train models effectively.

We discuss a weakly supervised training approach that collects only answer value annotations without rationales or equations in Chapter 7. To tackle this challenge, we present a novel search algorithm **ComSearch** which employs a combinatorial strategy for candidate equation searching. ComSearch effectively compresses the search space by excluding mathematically equivalent equations, enabling the algorithm to enumerate all possible equations and obtain high-quality training data. Additionally, we address the issue of the false-matching problem, which refers to the noise in pseudo labels that hold incorrect mathematical logic. To denoise the pseudo labels, we propose a ranking model that improves the quality of the training data.

Chapter 3

RODA: Reverse Operation based Data Augmentation for Solving Math Word Problems

3.1 Introduction

Given the complexity of mathematical equivalence and the challenges in annotating high-quality samples, expert annotation becomes necessary, which significantly increases the cost of annotation compared to other NLP tasks. In order to mitigate these challenges and improve the quality of numerical reasoning models, it is crucial to leverage the limited annotated samples effectively. However, despite the efforts made so far, the available data size remains insufficient, and the performance of existing approaches has reached a bottleneck. As a result, the primary consideration for enhancing the performance of solving math word problems (MWP) is to explore methods to augment the existing data. By finding effective ways to augment data, researchers can overcome the limitations imposed by the scarcity of annotated samples and further improve the performance of MWP solving models. Figure 3.1 shows two examples which include math word problems and their corresponding solution equations and results.

Data augmentation for NLP has been a critical and challenging research topic,

MWP1	MWP2
The distance between city A and B is **660 km**, the car starting from **A** drives **32 km/h**, and the car starting from **B** drives **34 km/h**. The two cars are starting from the two places **at the same time heading toward each other**. How many hours later would the two cars **meet**?	The car starting from A drives 32 km/h, and the car starting from B drives 34 km/h. The two cars are starting from the two places at the same time heading toward each other. *10 hours later the two cars would meet. What is the distance between city A and B?*
Equation: $x = 660/(32 + 34)$	Equation: $x' = 10 * (32 + 34)$
Knowledge: Time = Distance / Speed	Knowledge: Distance = Time * Speed
Answer: 10	Answer: 660

Figure 3.1: Two MWP Examples. The solution equations and mathematical knowledge of MWP1 and MWP2 are reversed to each other.

especially in the case of MWP solving. Traditional data augmentation methods are mainly based on paraphrase generation, such as EDA [106], which randomly edits words, and back translation [112], which translates the example to an auxiliary language and then back to the original language to paraphrase the example. These methods have two key drawbacks when applied to MWPs.

First, due to the preciseness of mathematics, the text description of each math word problem must be absolutely rigorous, so that even missing only one keyword could make the information incomplete and the problem unsolvable. As shown in Figure 3.1, all key information points of the problem marked in bold cover nearly one-third of the text content. With any of the key points missed, the problem would become meaningless. Therefore, in the task of MWP solving, paraphrase-based methods may potentially produce noise, mislead the model and degrade the performance. Second, the critical difficulty of MWPs solving lies in linking mathematical knowledge points and reasoning logic with examples. Here the mathematical knowledge points refer to formulae that can solve the MWP, such as $area_{circle} = pi * radius^2$, while the reasoning logic refers to the abstractive understanding of the relationship between these concepts, which is the desired ability of the MWP solving model. Paraphrase-based methods only introduce lexicon level variance but do not generate new examples on the mathematical level.

Therefore, the performance improvement of paraphrase-based data augmentation methods in the task of MWP solving is limited. Recently, a new research line of logic-based data augmentation uses task-specific logic to construct new examples for Natural Language Inference and Question Answering [2,34,46]. These methods introduce reliable rule-based logic, which does not produce noise, to generate logic-level new examples. However, existing methods are task specifically designed and cannot be directly applied to MWP solving. Meanwhile, these methods are based on one or two-step simple logic operations and are relatively simple.

In this chapter, unlike the previous practice of data augmentation, we instead simulate human double-checking and propose an MWP generation method to obtain more high-quality MWPs that are inferred through the reverse operation of the original problems. As we observe, when humans solve MWPs, a common technique to guarantee accuracy is to perform double-checking on the problem. As shown in Figure 3.1, MWP1 asks about the time that the two cars have spent before meeting each other, and its solution implicitly holds the mathematical knowledge point that $time = distance/speed$. With this knowledge, we can get the equation $x = 660/(32 + 34)$, where 10 is the final answer of x and denotes the spent time. To verify the correctness of the solution of MWP1, humans conceive a reverse problem MWP2 with the mathematical knowledge point $distance = time * speed$, which takes the answer of MWP1 (i.e., $x = 10$) as a known quantity and the distance x' as unknown. We can efficiently produce a new math word problem in the reverse operation and ensure its solution quality. Concretely, we can seek one known quantity (e.g., 660 in MWP1) in the original problem and change its surrounding declarative description into an interrogative sentence (i.e., the last sentence in MWP2). At the same time, we change the original interrogative description about the unknown quantity into a declarative statement with the original solution (e.g., $x = 10$) substituted (i.e., the next to last sentence in MWP2). Then, with most content unchanged, we can obtain a new math problem (e.g., MWP2) from the original problem (e.g., MWP1).

We can see that this kind of reversion-based data augmentation has the following benefits: First, this way of generating new data is relatively simple and reliable so that the key information will not be lost; Second, the reverse operation can infer

new knowledge points which helps to learn mathematical reasoning logic; Third, more high-quality data can be used to train the neural networks well. Next, we combine this Reverse Operation based Data Augmentation (RODA) with seq2seq models and conduct experiments on Math23K, the most influential large-scale dataset for MWPs. To be noticed, our method could be easily adapted to any supervised model on high-quality arithmetic MWP corpora with equation solutions. Experimental results show that our method could benefit various models and outperform previous state-of-the-art results on solving MWPs.

In summary, our contributions are three-fold:

- We follow the double-checking mechanism and propose the reverse operation based data augmentation method, which is easy to apply and accurate.
- We show how our method can introduce new mathematical knowledge points to the new examples and help to learn the reasoning logic.
- We effectively use data augmentation results in the MWP solving and achieve the state-of-the-art on the Math23K dataset.

3.2 Related Work

3.2.1 Math Word Problem Solving

3.2.2 Data Augmentation

The two most popular methods for sentence-level data augmentation in NLP are back translation [112] and EDA [106]. Yu et al. [112] used a high-quality machine translation system to translate the original text into a new language and then backward to perform paraphrase style data augmentation. Wei and Zou [106] slightly modified the input sentence on the token level by performing four kinds of operations: synonym replacement, random insertion, random swap, and random deletion. The drawback of applying these two methods on MWPs is that MWPs are very sensitive to even slight modifications; any key information missed the problem would become unsolvable. In addition, these methods only provide lexical level variance but no new reasoning knowledge.

Recently due to the reliability of rule-based data augmentation in NLP, task-specific logic rules for data augmentation have been explored in Natural Language Inference (NLI) and Question Answering. Kang et al. [46] studied NLI-specific logic-based data augmentation, which generates new examples by replacing tokens or changing labels on the original training examples.They only used the logic operations of NOT($\neg$) and equivalence($\Leftrightarrow$). Asai and Hajishirzi [2] further extended entailment($\rightarrow$) operation for common domain question answering data augmentation by measuring the transitive consistency of pairs of questions. Gokhale et al. [34] studied disjunction($\vee$) and conjunction($\wedge$) operation for yes-no style visual question answering. All these studies involve only one or two steps of simple logic. By contrast, our method uses reversed operation of complex mathematical computation and can introduce new reasoning logic in the generated new examples.

3.3 Methodology

Given a high-quality MWP dataset, we propose RODA, producing new accurate math word problems to enlarge the data scale. There is a linear correlation between the number of new questions generated and the number of known variables in the question. Next, we can use the augmented dataset to improve the supervised MWP solving models.

3.3.1 Reversion based Data Augmentation

To perform data augmentation, we reverse the original problems in the dataset to new problems, and the reversion process consists of three steps: number identification, problem transformation, and equation generation. To ensure the quality of the new data, the main criteria of our reversion process is "quality first quantity second", so that our method relies on some well-designed empirical rules in the three steps.

Number Identification

This stage consists of two steps. We first use an LSTM classifier to perform significant number identification and determine irrelevant numbers. Then we further filter out numbers that cannot perform reversed-based operation and then use exact match to map the numbers in the equation to the numbers in the question text. The statistics of this stage are given in Table 3.4.

A MWP might contain various numbers that are irrelevant to the solution, such as the date or description text such as 'tenth grade student'. Following Wang et al. [104]'s work, we perform significant number identification by building an LSTM-based classifier to determine the significance of the numbers and filtering out the irrelevant numbers. The classifier uses single layer LSTMs with 128 nodes and a symmetric window of length 3. The classification performance can reach around 99% accuracy.

In addition, the numbers in a math word problem do not necessarily map one to one with the numbers in the corresponding solution equation. One number may appear in the solution equation but do not appear in the word problem, and vice versa. Thus, to conduct high-quality word problem reversion, we first need to identify the valid numbers which can be converted to an unknown quantity using the reverse logic of the original solution. Here, we propose a rule-based method to identify the possible numbers for reversion. Four key rules are explained as follows, and we also show their examples in Table 3.1.

1. **Problem Number Duplication** If a number appears more than once in the problem, we filter this number out because we cannot map the numbers in the text to the equation with an injective function. As shown in Table 3.1, we cannot know whether a separate '2' is related to A or B, and thus it is difficult for us to conduct a precise reversion.

2. **Equation Number Duplication** If a number appears more than once in the equation, we filter this number out because it may not be capable of being solved with a linear equation with one unknown variable. To solve higher-order polynomial functions, introducing new operators such as root operation would be essential. However, such operators are out-of-domain

Rule	MWP examples	Num	Equation
1	A has 4 piles of 2 apples and B has 2 apples. B gave 1 apple to A, how many does A have in total now?	2	$x = 4 * 2 + 1$
2	The side length of a square is 2, what is the area?	2	$x = 2 * 2$
3	The side length of a cube is 4, what is the volume?	4 & 3	$x = 4^3$
4	The diameter of a circle is 5, what is the perimeter?	π	$x = \pi * 5$

Table 3.1: Examples of In-valid Numbers.

(OOD) with the original data and would introduce noise.

3. **Power Operation** If a number is involved with power operation, we filter it out because the inverse operation, which is logarithmic operation or root operation, is OOD. We filter out both the base number and the exponent number.

4. **Constant Term Numbers** Constant term numbers are not applicable for reverse operation, such as π and unit conversion terms.

We then perform exact match between the numbers in the question and the equation to align the numbers.

Problem Transformation

After collecting a set of valid number candidates, we perform problem reversion for each number to get a new transformed math word problem. The main work is to convert the original question sentence into a declarative sentence with a definite quantity and convert the sentence with the identified number into a question sentence with its question point on the number. Specifically, for the first conversion, we name the original question sentence as Q. From Q, we find the interrogative pronoun according to a list compiled in advance and replace the pronoun with

the answer of the original problem. Next, we adjust the word order to make the sentence fluent and natural and get the declarative sentence D'.

For the second conversion, given the candidate number c_i, we get the sentence D, which contains c_i. Next, we take c_i as the question focus and change D into a question sentence Q'. For different languages, the conversion process is different. For example, for Chinese, the conversion is relatively simple and just replaces c_i with an interrogative pronoun such as "多少 (How many)" without the need of adjusting word order.

We show the list of interrogative pronouns used for question conversion here in Table 3.2. Some of the interrogative pronouns also have other meanings in the declarative discourse unit, so we only detect them if they are in the last discourse unit of the sentence.

Chinese	**English Translation**
多少	How many/much
几分之几	the fraction number is
几	How many
=†	=
求	Please solve
((())/(()))	((())/(()))
多†	how much more than

Table 3.2: A list of interrogative pronouns. † denotes this pronoun is only valid when it is in the last discourse unit of the question.

For English problem transformation, we follow the manually encoded transformation rules from Heilman et al. [40]. We extend the interrogative pronoun candidates to the following list: *how many, how much, how far, how tall, how long, how fast, how old, how big, what fraction, what.*

We then edit the original math word problem. We delete the two sentences D and Q, and add D' and Q' at the end of the text. Then we get the new transformed math word problem. Our problem transformation method is simple but effective, which is the basis of producing new high-quality MWPs.

Equation Generation

For each new transformed word problem, we need to generate its solution equation. Similar to problem transformation, we derive the new solution equation according to the solution equation of the original problem.

To make the process of generating the new equation clear, we take the two MWPs in Figure 3.1 as examples. For the pre-identified number 660 in MWP1, it will be changed into a variable (i.e., x'). At the same time, we substitute the original variable x with its answer 10. Then we can get an intermediate equation $10 = x'/(32 + 34)$.

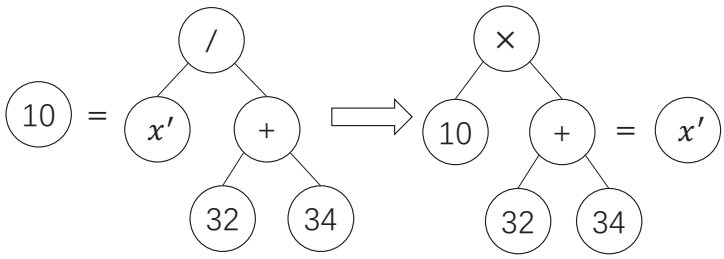


Figure 3.2: Example of equation conversion.

Algorithm 1 The algorithm of equation conversion

Input: Left part of medium equation l_tr and right part of equation r_tr

Output: Inverse equation in tree structure

1: $root = r\_tr.root$
2: $num\_tr, var\_tr = find\_var(r\_tr)$
3: **while** $root! = x^{'}$ **do**
4: $\quad l\_tree = rule(l\_tr, num\_tr, root)$
5: $\quad r\_tr = var\_tr$
6: $\quad root = r\_tr.root$
7: $\quad num\_tr, var\_tr = find\_var(r\_tr)$
8: **end while**
9: **return** $l\_tr = x^{'}$

Next, we need to convert this equation to its equivalent format where the

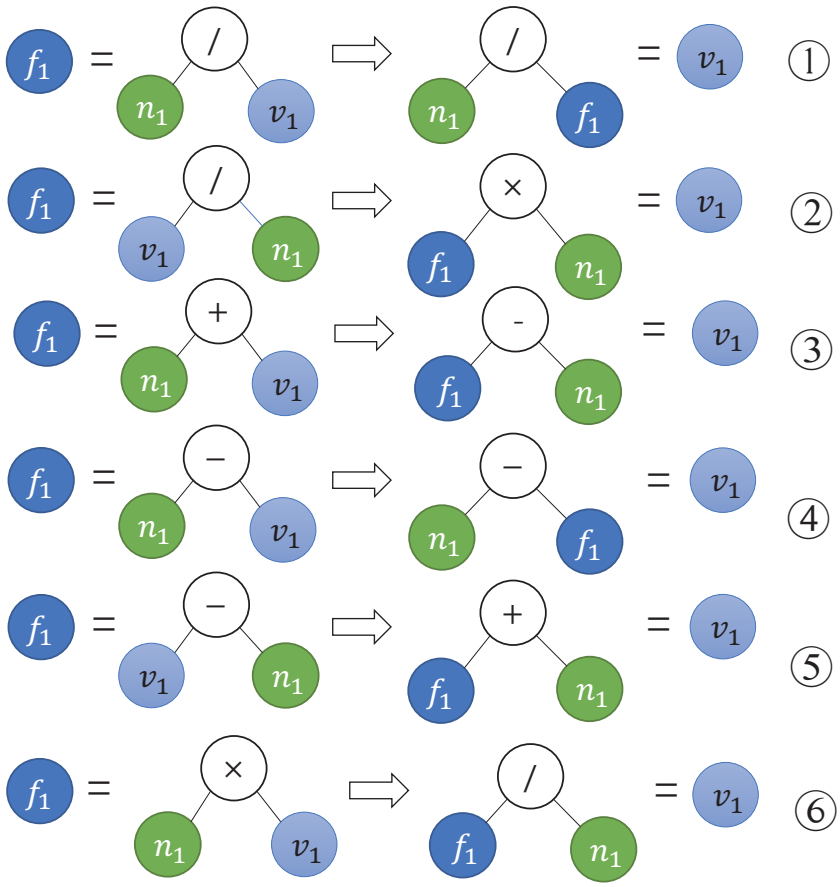


Figure 3.3: Rules of equation reversion

variable x' is located on the right side of the equal-sign alone. Figure 3.2 displays the equation conversion result of our running example. To conduct equation conversion, we design a recursive conversion algorithm based on the syntax tree structure. We first construct a quasi-binary syntactic tree for the original math equation. We denote the part on the left to the equal sign as f_1(the formula in stage 1). For the part right to the equal sign, we build a binary tree with one operator op_1 as the root node with two child trees. The child tree which has the new variable x' is marked as v_1(the child tree with a variable in stage 1) and the other one as n_1(the child tree without any variables in stage 1), and this identifying process is named as function *find_var* in Algorithm 1. This step

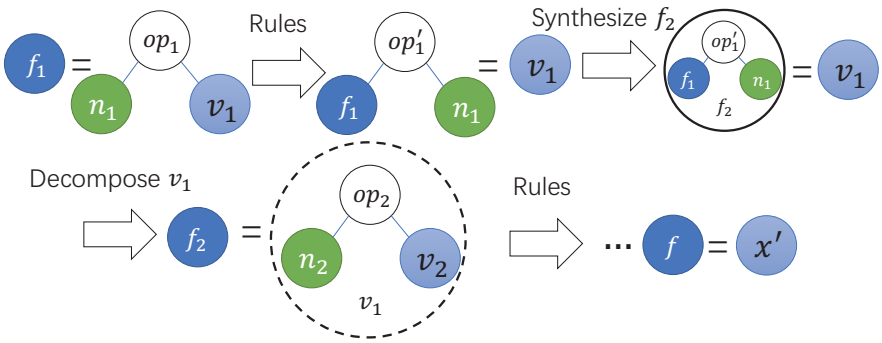


Figure 3.4: Illustration of recursive equation conversion.

corresponds to the upper left part of Figure 3.4. Next, like the upper middle part of Figure 3.4, we move n_1 to the left side and get a new operator op_1' according to equation reversion rules, which are summarized from the basic mathematical computation. We show thee rules in Figure 3.3. Regarding f_1 and n_1 as two new child trees and op_1' as the new root node, we can get a new tree f_2, which is the black circular ring in the upper right part of Figure 3.4, and we just use a single node f_2 to represent it in the following step.

Then, v_1 is further broken down as a binary tree which is composed of the root node op_2 and the child trees n_2 and v_2 as shown in the lower-left part of Figure 3.4. If we ignore the dotted circular sign of v_1, in this state, the equation has the same structure as the beginning state, so the following process will repeat until v_n has only one node x'. At this time, referring to the lower right part of Figure 3.4, all nodes of numbers and operators are moved to the left side and only the variable node x' is left on the right side. The concrete process of equation conversion is shown in Algorithm 1.

Since one math equation can be written in several equivalent forms (e.g., $4+2-3$ and $4-3+2$), which brings noise to the model training, we conduct equation normalization for all the original and generated equations. We follow the criteria of Wang et al. [102] that if one equation could be converted into a shorter one, then it should be shortened, and the order of the numbers in one equation should follow their occurrence order in the problem text as much as possible. Wang et al. [102] performed equation normalization by applying a series of rules

that only work on limited cases such as a sequence of multiplication. Here we increase the applicable scope. We also use the simplification algorithm supported by sympy [72] that heuristically simplifies the equations and matches with human writing regulations. We show further details of the algorithm in the appendices.

This equation normalization is essential for assuring the model performance since the equations generated by the reverse operation are written in a completely different style from that of the equations in the original data. The normalization can avoid domain shift of the prediction target equations.

To be noticed, sympy would put negative numbers in the first position of one equation during simplification, which causes the problem that one equation no longer can form a binary abstract syntax tree. We also design an algorithm to avoid this problem by moving the first addition term to the first position. We show an example in Table 3.3.

Step	Equation
Equation	$x = c - a - c + (c * a) + (b/b)$
Sympy	$x = -a + 1 + (a * c)$
Adjusted	$x = 1 - a + (a * c)$

Table 3.3: One example of equation normalization

We show the algorithm for removing the negative terms in the front of the equation during equation normalization in Algorithm 2. To be noticed, this process does not change the mathematical value of the equation but only how it is written.

3.3.2 MWP Solving Model

Our data augmentation method can be combined with any preferred neural model. Here, we adopt two seq2seq models and one classification model, then examine their performance. First, we implement Liu at el. [65]'s prefix model, which is a light and effective baseline. At the same time, to show whether an advanced model can be further improved by our augmented data, we use the SOTA model named Goal-driven tree-structured MWP solver (GTS) [109], which is an exten-

Algorithm 2 The algorithm of removing negative terms

Input: Original Equation

Output: New Equation Written Form

```
while True do
  if Brackets in Equation then
    subs = list of sub-equation in the brackets;
    for all sub-equation in subs do
      Remove Negative(sub-equation)
    end for
  end if
  if Equation[0]=='-' then
    Find the first add token;
    Move the token to the front;
  end if
end while
return Equation
```

sion of the prefix baseline model. We also build a classification model based on Transfromers following Kushman et al. [56] and Robaidek et al. [89], which considers the equation template as the label of one MWP example. Here, we briefly introduce the three models.

For the prefix model, formally, the model takes a sequence of tokens $\{x_i\}_{i=0}^{n}$ as the input and embeds them into a sequence of word embedding representations $\{e_i\}_{i=0}^{n}$ which are fed into a bidirectional long short term memory network (BiLSTM) encoder. Then two context-aware representations $h_i^{enc_f}$ and $h_i^{enc_b}$ are calculated and concatenated as h_i^{enc} for each token. Then these representations are given to the decoder to decode the output equation.

The decoder adopts a unidirectional LSTM to generate the output in an autoregressive manner. At each decoding time step t, the decoder calculates the attention weight distribution $\{a_t^i\}$ on $\{h_i^{enc}\}$ with the embedding e_t^{dec} of the output of the previous time step y_{t-1} and the current hidden state h_t^{dec} of the decoder LSTM, and assigns them to the encoder outputs $\{h_i^{enc}\}$ to form an attention-aware

representation s_t which is finally fed to a Multi-layer Perceptron (MLP) layer to generate the output token y_t.

$$h_t^d = LSTM(h_{t-1}^{dec}, e_t^{dec}) \tag{3.1}$$

$$\begin{aligned} s_t &= \sum_{i=1}^{n} \alpha_t^i \cdot h_i^{enc} \\ &= \sum_{i=1}^{n} \frac{exp(h_i^{enc} \cdot h_t^{dec})}{\sum_{j=1}^{n} exp(h_j^{enc} \cdot h_t^{dec})} \cdot h_i^{enc} \end{aligned} \tag{3.2}$$

The GTS model [109] further extends the prefix baseline with subtree representations which can provide more information for the decoding process. A recursive neural network is used to encode subtrees of the equation in a bottom-up manner. The subtree representation of one token y_t is calculated based on its children nodes with the gate mechanism. With the subtree representations, this model can also well use the information of the generated tokens to predict a new token.

It is noted, for the seq2seq models, directly generating the solution equation with numbers suffers from a serious out-of-vocabulary (OOV) problem since the vocabulary of numbers is enormous. To address this problem, we follow Kushman et al. [56], which used equation templates instead of actual equations as the prediction target of the model. The numbers in one MWP are notated as $temp_i$, where i denotes the order of the numbers that appear in the problem. Extra constant numbers such as π and 1 are also added to the decoder vocabulary. Then the OOV problem can be solved.

For the classification model, we follow Kushman et al. [56] and Robaidek et al. [89], which encodes the question text and then classifies the corresponding equation template. We add a [CLS] token to the question text and use a Transformer to encode the question text. We feed the feature vector of the [CLS] position to a multi-layer perception network to classify the equation template.

To further improve the models, we mix and shuffle the augmented data with the original data as the new training set of the models. It is noticed that our method does not limit to these three models.

3.4 Experiments

3.4.1 Experiment Setup

In our experiments, we mainly experiment on Math23K[1] which is the most influential large scale dataset for MWP solving in the Chinese language. It contains 23,162 one unknown variable elementary school level MWPs with the corresponding equation solutions. In train-test setting, the dataset is splitted to 21,162 training examples, 1,000 validation examples and 1,000 test examples.

For the classification model, we use Transformer base with 12 layer, 768 hidden state size and 12 attention heads. The classification model has 5693 labels. We train with a 16 batch size for 20 epoches. We use the SGD optimizer with a learning rate of 2e-5 and weight decay of 0.01.

For the two seq2seq MWP solving models, we fix their embedding size to be 128. For the prefix baseline model, the dimension of encoder hidden state is set to 512 while the dimension of decoder hidden state is 1024. For the GTS model, the hidden state of both encoder and decoder is 512. We use Adam optimizer to optimize these parameters. The batch size is 128. To compare the performance with baselines fairly, GTS model is tested by both train-test setting same as the other models, and also 5-fold cross validation setting same as the original chapter[2], while others are tested on the test set[3]. The experiments are done on GTX 1080Ti GPU, with a runtime of 10 hours for prefix baseline, 110 hours for GTS and 15 hours for the classification model.

For data augmentation, we perform the reverse operation on the training set of Math23K, which includes 21,162 MWPs. From these problems, we can get 58,699 numbers by using the LSTM-based classifier. At the same time, we filter out 1,490 problems which are composed of only numbers and operators such as 'Please calculate 5+7*10', because they can not give effective supervision to the MWP solving models. We also exclude 7,399 numbers in the problems which are not easy to be reversed, because they do not map one to one with the numbers

[1]Download link: https://github.com/SumbeeLei/Math_EN

[2]We filter out the augmented samples of test set for each cross. So in the training stage, models can only learn from the training set and their augmented samples.

[3]We use the same split as Wang et al. [102]. Only training set is used for augmentation.

Type	#	Prop.
Original Problems	21,162	-
Filtered Problems	1,490	0.07
Original Numbers	58,699	2.77
Candidate Numbers	54,717	2.59
Irreversible Numbers	7,399	0.35
Augmented Problems	47,318	2.24

Table 3.4: The statistics of the data augmentation on the training set of Math23K. **"Prop."** stands for the proportion of the item compared with original problems.

Type	#	Prop.
Dev Set Templates	377	-
Original Covered	307	81.4
+ RODA	322	85.4

Table 3.5: The statistics of the template coverage of Math23K train set templates on development set. **"Prop."** stands for the proportion of the item compared with original problems.

in the solution equations as stated in Section 3.1. Finally we totally get 47,318 new problem-equation pairs. We can see, our data augmentation method successfully generates new data whose size is more than two times the original data, demonstrating the method's ability for performing large-scale data augmentation cheaply. We illustrate the statistics of the data augmentation results in Table 3.4.

We also show the statistics of template coverage on the development set of the original training set and the augmented training set in Table 3.5. As we can see our data augmentation method increased the template coverage of the development set, reducing the rate of uncovered templates by 21.5%. This can show how our method introduces new mathematical knowledge points and benefit the model.

Model		**Acc**
Retrieval	Cosine [89]	23.8%
	Jaccard [89]	47.2%
Classification	Transformer [89]	56.8%
	Bi-LSTM [89]	57.9%
Generation	DNS [104]	58.1%
	BiLSTM+Suffix+EN [102]	66.7%
	TreeLSTM [65]	69.0%
	Group-Attention [58]	69.5%
Ensemble	DNS+Retrieval [104]	64.7%
	DNS+suffix+EN Ensemble [102]	68.4%
Classification	Transformer Ours	54.9%
	Transformer Ours+RODA	63.7%
Generation	Prefix [65]	67.8%
	Prefix+RODA	**70.5**%
	GTS [109]	75.6%
	GTS+RODA	**77.9%**
	GTS† [109]	74.3%
	GTS+RODA†	**76.0%**

Table 3.6: Math word problem solving accuracy on Math23K. † denotes that the result is 5-fold cross validation performance. All other models are tested on the test set.

3.4.2 MWP Solving Results

We first evaluate the MWP solving performance by comparing our methods with other baseline methods. Here we name our Reverse Operation based Data Augmentation method RODA. In this experiment, we use all the 47,318 augmented data for training the prefix and GTS models. Table 3.6 shows the results of our methods and other novel systems on the Math23k evaluated by the final answer accuracy. In this table, we classify the MWP solving models as retrieval-based, classification-based, generation-based and ensemble models. The retrieval-based

models mainly calculate a similarity score for questions in the test set and the questions in the training set and assign the template that has the highest similarity [89, 101]. **Cosine** and **Jaccard** respectively denote the methods which adopt the corresponding similarities. The classification-based models train a classifier to predict an equation template for each problem in a multi-class classification manner [56]. For retrieval and classification models, we use the results from Robaidek et al. [89].

The generation-based models are the recent mainstream for MWP solving and use end-to-end seq2seq models to directly generate an equation template. Wang et al. [104] proposed the **DNS** model, which used seq2seq with significant number identification to generate an equation template. Wang et al. [102] improved their model and proposed the **Suf+EN** model, which extends the DNS model by decoding the suffix notation and performs equation normalization for preprocessing. **TreeLSTM** [65] uses a top-down tree-structured decoder to predict the equations. **Group-Attention** [58] uses various attention methods to capture the intra-relation of the numbers. We also use two ensemble models **DNS+Retrieval** and **Suf+EN Ensemeble** for comparison, which uses bagging to combine the results of different models.

Our data augmentation method is exerted on two generation models Prefix [65] and GTS [109] which have been introduced in Subsection 3.2. We choose these models as they can somewhat be representative of generation-based methods, especially GTS achieves the SOTA performance. From Table 3.6, we can see that generation-based methods generally outperform retrieval-based and classification-based methods. To show the efficiency of our method, we conduct experiments on both the classification models and the SOTA generation models. We can see that our method gains 8.8 points of improvement over our Transformer baseline classification model. We also show that our RODA method can further promote the current SOTA models: Prefix+RODA and GTS+RODA boost Prefix and GTS by about 2.7 points and 2.3 points, respectively. Under 5-cross validation setting, our model also improves GTS by 1.7 points. This also exhibits that more high-quality data is useful for improving the performance of MWP solving.

Data	Data Prop.	Acc.
Original only	–	68.0%
RODA Only(2.24 times)	–	50.0%
Orig+RODA	1:0.5	69.5%
Orig+RODA	1:1	69.8 %
Orig+RODA	1:1.5	70.9 %
Orig+RODA(All)	1:2.24	**71.0%**

Table 3.7: Effects of data augmentation by adjusting the augmentation proportion on the devlopment set. The middle column denotes the proportion of original data and augmented data.

3.4.3 Analysis of Data Augmentation

Here we further investigate how the augmented data improves the MWP solving model. In consideration of the balance of experiment time and accuracy, we use the Prefix model on the development set of Math23K for analysis.

Previous studies on data augmentation [112] show that too much augmented data might harm the performance of the model. Thus, we experiment with what percentage of our augmented data can best improve the MWP solving model. We only use our augmented data to train the model and achieve the accuracy of 50%, still far lower than only using the original Math23k training data (68%). The cause of this low performance could be caused by various reasons. First, the number identification stage filters out various types of questions, that the model can not deal with these unseen examples. Second, the new examples might introduce noise, that the augmented problem is similar to the original one in lexical terms and is hard to differ for the solver. Third, while our method can cover more mathematical knowledge points, it does not completely match with the original distribution. For example, considering the rectangle circumference formula $C = (a + b) \times 2$, the reversed formula $a = C/2 - b$ would appear less in the problem distribution. We leave these problems for future work.

We also consider combining the original data with different proportions of augmentation data as training data whose performance is shown in Table 3.7.

Model	Acc
BT Only	45.6 %
RODA(1:1) Only	48.2%
RODA(1:2.24) Only	50.0%
Origin	68.0%
+ BT	68.2 %
+ RODA(1:1)	69.8%
+ Full RODA	**71.0%**

Table 3.8: Comparison with Back Translate data augmentation method.

We can see that, as the size of the augmented data increases, the performance stably increases, which shows how the size of the training data affects the performance. The model performs similarly when the proportion is 1.5 and using the whole augmented data. Because the performance of the model has not decreased along with the increase of augmented data, we use the full augmented data for the reported results. This can demonstrate the reverse operation can infer new knowledge points, which helps to learn the mathematical reasoning logic while more high-quality data can be used to well train the neural networks.

We also compare our data augmentation method with another novel data augmentation method, back-translate (BT) [112] . For the BT method, we translate the problem text into English and then back to Chinese by Google Translate[4]. As BT can only perform data augmentation with the 1:1 proportion of the original data, we also control the size of our augmented data. Table 3.8 compares the MWP solving results. As shown in Table 3.8, We can see that the performance of BT has a significant gap with RODA, even when the data augmentation proportion is the same. There are two reasons for such a performance gap. First, BT may introduce more noise into the training data through the translation-based paraphrase and degrade the model performance, while our method can better ensure data quality. Second, BT only paraphrases the same meaning on the lexicon level, while our data augmentation method can introduce new knowledge points

[4]https://translate.google.com/

#	**Chinese Text**	English Translation	Equation	Coh	Cor
1	甲,乙两同学相距 t_a 米,同时相向而行,乙同学速度为t_b 米/秒,与此同 时,一只小狗以 t_c 米/秒的速度,从甲身 边跑向乙,遇到乙后又以同 样的速度 跑向甲,…如此往返,直到甲、乙同学相遇,问在此段 时间内,小狗共跑了 t_d 米,甲同学速度为多少米/秒?	The distance between A and B is t_a meters. The two are heading toward each other. The speed of B is t_b m/s. Meantime, a dog starts with A and then runs back and forth between the two people with a speed of t_c m/s, until the two people meet. During this duration, the dog ran t_d meters. What is the speed of A?	$x = t_a * t_c/t_d - t_b$	5	1
2	几个 小朋友 分 苹果 , 如果 每人 分 t_a 个 , 如果 每人 分 t_b 个 , 少 t_c 个 , 小朋友 有 t_d 人 , 就 余 多少 个?	A few children are sharing apples. If each child gets t_a apples. If each child gets t_b apples, there would be a shortage of t_c apples. There are t_d children. How many apples are left?	$x = t_d * (t_b - t_a) - t_c$	2	0

Table 3.9: Examples of the output from the Data Augmentation Module. **"Coh"** and **"Cor"** stands for Coherence and Correctness in Table 3.11.

that helps to learn the mathematical reasoning logic.

3.4.4 Human Evaluation

To further examine the data quality of our augmented data, we perform human evaluation to examine the original data, back-translate augmented data, and RODA data. We sample 100 MWPs from each dataset. 25 examples have two numbers, 25 examples have three, 25 examples have four, and the last 25 have five or more numbers. The sampling in each category is random. The evaluation involves two aspects. The first is the coherence which is ranked between 1-5. This examines whether the generated text is coherent. The second is correctness which is classified as either 0 or 1. This examines whether the equation matches the problem text. Two annotators participate in ranking the data. Table 3.11 lists the average scores for each dataset with respect to the two metrics. From Table 3.11, we can see our method can generate new data which has only a small performance gap with the original data. Compared to the BT method, our augmented

Original Text	**BT**	**BT English**	**Origin & BT Equation**	**Coh**	**Cor**
某江汛期时中流与沿岸的水速有很大不同，中流每小时t_a里，沿岸每小时t_b里。今有一汽船顺中流而下，t_c小时行驶了t_d里，问从沿岸返回原处需几小时？	在河流中间，中游和海岸的水流速度非常不同。中游时速为t_a英里/小时，海岸时速为每小时t_b英里。今天，有一艘汽船沿着河流航行，在t_c小时内行驶t_d英里。从海岸返回原始地点需要多长时间？	In the middle of a river, the water velocity of the middle stream and seacoast are very different. The middle stream velocity is t_a miles per hour and the seacoast velocity speed is t_b miles per hour. Today, a steamer sails down the river and travels for t_d miles in t_c hours. How long does it take to return to the starting point from the seacoast?	$x = t_d/(t_d/t_c - t_a - t_b)$	2	0
Origin Text English	**RODA**	**RODA English**	**RODA Equation**	**Coh**	**Cor**
During the flood season of a certain river, the water velocity of the middle stream and the coastal waters are very different. The middle stream velocity is t_a kilometers per hour and the coastal velocity speed is t_b kilometers per hour. Now a steamer sails down the river in the midstream and travels for t_d miles in t_c hours. How many hours does it take to return to the starting point in the coastal waters?	某江汛期时中流与沿岸的水速有很大不同，沿岸每小时t_a里，今有一汽船顺中流而下，t_b小时行驶了t_c里，问从沿岸返回原处需t_d小时，中流每小时几里？	During the flood season of a certain river, the water velocity of the midstream and the coastal waters are very different. The coastal velocity speed is t_a kilometers per hour. Now a steamer sails down the river in the midstream and travels for t_c miles in t_b hours. If it takes t_d to return to the starting point in the coastal waters, What is the midstream velocity?	$x = t_c/t_b - t_c/t_d - t_a$	5	1

Table 3.10: Comparison of Back Translation and Reversed Operation Based Data Augmentation. **"Coh"** and **"Cor"** stands for Coherence and Correctness in Table 3.11.

data is of higher quality in both coherence and correctness. This can demonstrate how our method is reliable so that the key information is not lost, which is also why our augmented data can well boost the model performance.

Model	**Coherence**	**Correctness**
Original	4.27	0.92
BT	3.24	0.55
RODA	3.86	0.84

Table 3.11: Human evaluation on Math23K.

Here we show two augmented examples in Table 3.9. In case 1, this newly produced example has coherent text and correct solution, even if the original mathematical logic and description text is fairly complex. The original example involves the numbers that the time used by A, B, and the dog is the same so that it can form the problems for each of the 5 variables: the speed of A, B, and the dog, the distance between A and B and the distance that the dog has run. In this example, the original training data can be augmented into 4 new high-quality question-answer pairs, and each of them holds a new mathematical knowledge point, which demonstrates the effectiveness of our model. We also show an example where our method failed in case 2. When swapping the order of the sentences, the coreference resolution of **apples** in the final question has changed that it no longer asks about the origin variable; therefore, the new reversed question would no longer be natural nor correct. In the case that the sentence order swapping would influence the coreference resolution of natural text, our method would no longer work. Such error are caused since our question generation module does not consider the dependency between discourses. Such kind of error could be reduced with an additional discourse analysis module, which could be left for further work.

We show one example of comparison between BT and RODA in Table 3.10. As we can see, during the paraphrase BT translates 中流 into 中游 and 沿岸 into 海岸, which makes the question no longer natural and reasonable. Meanwhile, it leaves out a key information point 顺中流而下 that the question meaning has completely changed and the equation no longer matches with the question. Such kind of minor errors during the paraphrase would be fatal for MWP data aug-

mentation that the key information is changed. The noisy new examples would reduce the performance of the model. Meanwhile, our method RODA correctly formed a natural reversed question with a new mathematical knowledge point corresponding to the reversed equation.

3.4.5 A Study on English Dataset

Model	Acc
GTS	68.2%
+ BT	70.3 %
+ Full RODA	**70.7%**

Table 3.12: Results on AllArith.

We also extend our data augmentation method to AllArith[5] [91], which is a high-quality English MWP dataset with 831 problems, to show how our method works on an English dataset. As shown in Table 3.13, our data augmentation method generated 715 new examples for AllArith, respectively. We use the GTS model in this experiment and compare it with the BT data augmentation method. The results are based on 5-cross validation following the split of Roy and Roth [91].As we can see in Table 3.12, our model achieves performance improvement when adding the augmented data, which demonstrates the generalization ability of our method beyond language. Here the BT data also achieve performance improvement, slightly lower than using our augmented data. We consider the reason why our data augmentation method does not exhibit significant advantages over back translation for two reasons: First, the samples in Math23K have more variables in average (2.77 for Math23K and 2.35 for AllArith), that our method can generate more new augmented examples for Math23K; Second, the samples in Math23K are more difficult in mathematical reasoning complexity, our method can introduce new mathematical knowledge points that can benefit the model to learn mathematical reasoning logic. In the future, we will research how to produce more high-quality data.

[5]Download link:https://github.com/CogComp/arithmetic

Type	#	Prop.
Original Problems	831	-
Filtered Problems	1	0.00
Original Numbers	1953	2.35
Candidate Numbers	1951	2.35
Irreversible Numbers	1236	1.49
Augmented Problems	715	0.86

Table 3.13: The statistics of the data augmentation on the full set of AllArith. **"Prop."** stands for the proportion of the item compared with original problems.

3.5 Summary of This Chapter

In this chapter, we propose the reverse operation based data augmentation for MWP solving, which converts the question and equation via reverse operation. Enlightened by how humans perform double-checking during calculation, the method can perform cheap and accurate data augmentation that could be adapted to any model. The augmented data also provides supervision of new mathematical knowledge points that could benefit the model beyond paraphrasing the text. We evaluate our method on Math23K and achieve state-of-the-art performance. In comparison with a strong baseline Back Translation, we show how our method significantly outperforms on a complex and large-scale dataset Math23k and achieve comparable performance on a simple and small dataset AllArith.

Chapter 4

Comprehensive Solution Program Centric Pretraining for Table-and-Text Hybrid Numerical Reasoning

4.1 Introduction

Given the challenges associated with the complexity of mathematical equivalence and the high cost of expert annotation, researchers have been actively exploring techniques to improve numerical reasoning models. This study aims to address the challenges by leveraging limited annotated samples effectively and developing strategies to mitigate the high cost.

The field of natural language processing has seen a growing interest in developing techniques for Question Answering (QA) style numerical reasoning on both textual data [26, 44, 104] and tabular data [19]. Recent research has focused on addressing the challenge of numerical reasoning over Table-and-Text hybrid data [17], which is a rich area for applications such as financial analysis. As demonstrated in Figure 4.1, the pipeline [17] first uses a retriever to extract a subset of supporting evidence that contains the required variables from a long

table-and-text hybrid passage. Then, a sequence-to-sequence program solver is trained on the retrieved variables to predict the solution program, as shown in the example of '$divide(1760, add(279, 320))$'. This solution program can be represented as an abstract syntax tree, as illustrated in the right section of Figure 4.1.

Such retrieve-then-solve framework, while simple, has limitations that make it difficult for the model to learn the task effectively. One issue is that the retrieved evidence could be noisy and contain irrelevant variables, which hinders the performance of the program solver. As illustrated in Figure 4.1, the red-colored variables '170.1' and '7' are irrelevant to the question. Numerical reasoning demands high precision, and systems are sensitive to noise. Studies on adversarial attacks for numerical question answering [55, 81, 110] have shown that irrelevant information can mislead the model and harm performance. The model may struggle to select the correct variables for reasoning, leading to incorrect predictions. Another limitation is that the program generation task requires the model to perform multiple steps of sub-program construction to generate the final solution program tree, which is a challenging task for deep learning systems [111], especially considering the task involves table-specific operators that involve multiple variables such as *'table_average'*. Additionally, the system only receives coarse-grained supervision of the whole program during training, meaning it only knows the final program, not which evidence each sub-program is derived from. This makes it difficult for the model to learn about the underlying reasoning process and to generalize to new examples.

In this chapter, we aim to tackle these challenges by introducing a novel structured pretraining framework that effectively leverages annotated programs obtained from supervised data. Our proposed framework encompasses three pretraining tasks, from the whole program level to the sub-program level. Specifically, to enhance the model's ability to distinguish between required and irrelevant variables in retrieved evidence, we propose *Variable Integrity Ranking* pretraining task. As illustrated in Figure 4.1, given two evidence and question pairs that contain different sets of variables, such as orange *'set 1'* that contains all required variables and green *'set 2'* that only contains partial required variables, the model

Passage:
... We will continue to evaluate opportunities that arise , and we will utilize this strategy to increase our number of apartment communities in strong and growing markets . We acquired the following apartment communities during the year ended December 31, 2017:

Community	Market	Units	Closing Date
Charlotte at Midtown	Nashville, TN	**279**	March 16, 2017
Acklen West End	Nashville, TN	**320**	December 28, 2017

... We also consider portfolio dispositions when such a structure is useful to maximize proceeds and efficiency of execution. During the year ended December 31, 2017, we disposed of five multifamily properties totaling **1760** units and 4 land parcels totaling approximately 23 acres. ...
During the year ended december 31, 2017, we incurred $**170.1** million in development costs and completed **7** development projects. ...*(abbreviate ~10 sentences)*

Question: During the year ended December 31, 2017, what was the ratio of the units disposed to the units acquired?

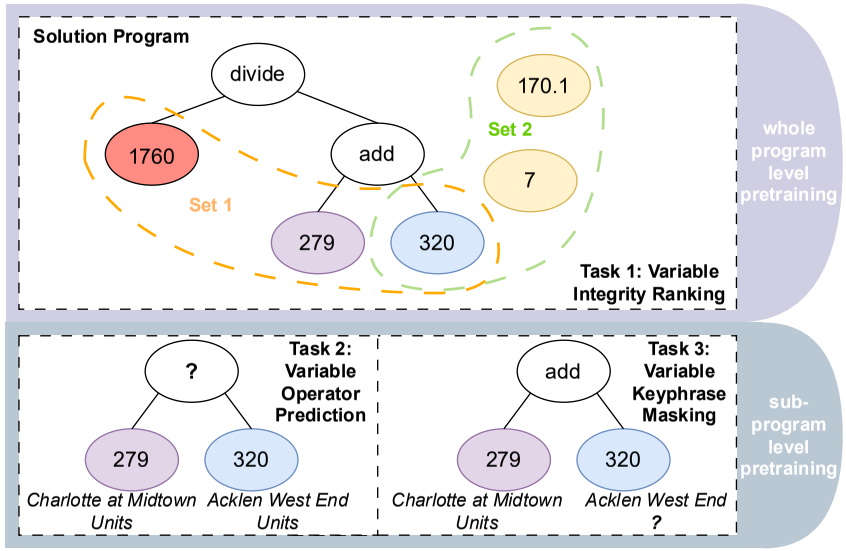


Figure 4.1: Example of Hybrid Numerical QA and our proposal. The yellow, purple and blue color denotes required variables for solving the question. The red variables denotes irrelavant variables.

is trained to rank which set of variables contains more required variables. The model can learn to eliminate irrelevant variables and focus on useful supporting evidence for reasoning.

In order to provide fine-grained supervision for the construction of sub-programs, we propose two novel pretraining tasks, namely *Variable Operator Prediction* and *Variable Keyphrases Masking*, which decompose the coarse-grained annotation into sub-program level. We observe that the variables in the target program can provide direct guidance for sub-program construction by breaking down the program into the smallest sub-program unit consisting of two variables and an operator. As demonstrated in Figure 4.1, given an evidence and question pair, the model is trained to predict the operator (e.g., *'add'*) between two variables (e.g., '279' and '320'), where the decomposition of the final program provides the supervision. This single-operator reasoning task helps the model learn the underlying reasoning of sub-program construction, thus allowing the model to perform more accurate numerical reasoning. We also observe that keyphrases such as technical terms and dates often serve as critical information for the construction of sub-program. As illustrated in Figure 4.1, given an evidence and question pair, the model is trained to recover the masked keywords describing the variables (e.g., *'Units'*) by considering other pieces of evidence and inferring which variable information is required to answer the question. This allows the model to understand how keyphrases information determines the sub-program construction.

In the remaining of the chapter, we introduce task settings and baseline model in Section 4.2, methodology of our three pretraining tasks in Section 4.3, experimental settings of the dataset, evaluation metric and implementation details in Section 4.4, results of applying the three pretraining tasks to existing transformer-based pretrained language models (PLMs), ablation study and case study in Section 4.5, and related works in Section 4.6.

4.2 Preliminary Background

As per the methodology outlined by Chen et al. [17], for each example in the training data, a hybrid table-and-text hybrid passage, a question Q, a solution

program P, and an annotation of the gold evidence set are provided. The cells in the table are transformed into sentences by concatenating the row and column headers using a template. For example, the purple cell labeled 'Units' in Figure 4.1 can be transformed into the sentence *'The Charlotte at Midtown of Units is 279'*. The cells within the same row are concatenated to form a single piece of evidence.

We leverage FinQANet [17] as the numerical reasoning model. It is composed of two main components: an evidence retriever that first retrieves the relevant evidence from the input financial report, followed by a program solver that generates the solution program.

Evidence Retriever The original hybrid passage forms an evidence candidate sentence set $S = s_1, ...s_n$, and the gold evidence forms a set of sentences $S^{gold} = \{s_1^{gold}, ...s_k^{gold}\}$. The objective of the evidence retriever is to assign a score to each sentence in S, which reflects the likelihood of it appearing in S^{gold}. Each candidate supporting evidence sentence s_i is concatenated with the question and then passed through a binary classifier. The top-n retrieved facts, as determined by the scores of the classifier, are reordered as they appear in the passage and utilized as input for the program solver.

Program Solver The program solver is an extension of sequence-to-sequence models, which takes the retrieved supporting evidence as the encoder input. The decoding target is a flattened representation of the solution program. For example, the program '$divide(1760, add(279, 320))$' is flattened to $add(279, 320), divide(1760, \#0)$, where $\#0$ denotes the first decoded sub-program $add(279, 320)$. To enhance the model's ability of capturing structural information provided by the decoding history of sub-programs, the representations of these sub-programs are also provided as input to the decoder, in addition to the encoder outputs.

4.3 Methodology

Our proposed approach applies the three pretraining tasks to a universal encoder for the three tasks. We then use the pretrained model to initialize the FinQA re-

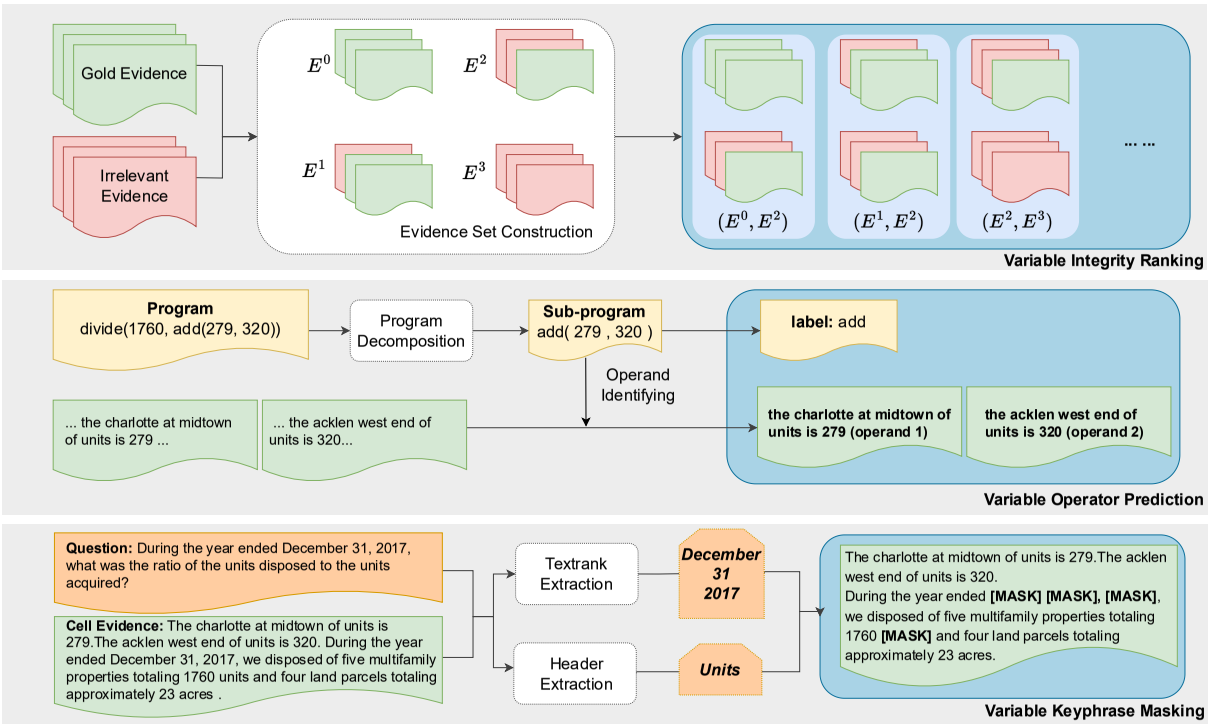


Figure 4.2: Pipeline of the construction of pretraining data.

triever and solver encoder for task-specific fine-tuning. The construction pipeline of the pretraining task data is depicted in Figure 4.2. In Section 4.3.1, we introduce *Variable Integrity Ranking* (**VIR**), which utilizes whole program level supervision to guide the model to contextually learn which variables are required. In Section 4.3.2 and Section 4.3.3, we present two sub-program level pretraining tasks: *Variable Operator Prediction* (**VOP**), which leverages fine-grained supervision to enable the model to learn single operator reasoning, and *Variable Keyphrase Masking* (**VKM**), which provides immediate guidance for the model to benefit from the reasoning of using keyphrases to construct sub-programs. We perform the pretraining using a multi-task training strategy, which is described in Section 4.3.4.

4.3.1 VIR: Variable Integrity Ranking

This pretraining task aims to train the model to rank the number of gold evidence pieces in a question and evidence sentence, i.e., variable integrity levels, aiming to improve the model's ability to distinguish between required and irrelevant variables. In order to construct examples with varying integrity levels, as depicted in Figure 4.2, a bag of sentences is selected from both gold (green) evidence and irrelevant (red) evidence within the passage to form pseudo-evidence sets E^i, where the integrity level i signifies the number of irrelevant evidence included. Consequently, smaller i sets correspond to higher integrity. These generated sets are subsequently utilized to train the model in the learning-to-rank framework.

Specifically, given an example from the training data, the gold evidence set E^0 includes all the required variables and is thus defined as the level 0. To construct subsequent evidence sets, one gold evidence s_j^{gold} is selected from E^i, and replaced with irrelevant evidence randomly drawn from the original passage S. This replacement process is repeated k times, resulting in the collection of $k+1$ pseudo evidence sets $E^0, ...E^k$. These sets are subsequently utilized for training the model in the learning-to-rank framework by creating evidence set pairs, denoted as $(E^u, E^v)|_{v=u+1}^{k}|_{u=0}^{k-1}$, yielding a total of $(k+1)*k/2$ pairs for pretraining.

In order to rank the integrity of variables, we employ a pairwise learning-

to-rank algorithm, RankNet [10]. The evidence set pair (E^u, E^v) is separately concatenated with the question text Q and fed into a tranformer-based PLM *Encoder*. The representations (h^u, h^v) of the example are obtained by utilizing the *[CLS]* token representations. Then, the representations are mapped by one layer of feed-forward network (FFN) and activation function *tanh* to a single dimension score (s^u, s^v), where a higher score stands for higher variable integrity level.

$$\begin{aligned} h^u &= Encoder_{[CLS]}(Q; E^u) \\ s^u &= tanh(FFN(h^u)) \end{aligned} \tag{4.1}$$

The final loss is calculated by Binary Cross-Entropy over the subtraction of the two scores:

$$\mathcal{L}_{rank} = BCE(s^u - s^v) \tag{4.2}$$

4.3.2 VOP: Variable Operator Prediction

The proposed model is trained to predict the operator between two operands within a given question and evidence sequence. As illustrated in Figure 4.2, the program $divide(1760, add(279, 320))$ can be decomposed into sub-programs $add(279, 320)$ and $divide(1760, \#0)$. The latter sub-program contains an intermediate operand, for which the representation does not explicitly exist in text. Thus to construct the training examples, the original solution program is decomposed, and the sub-programs in which both operands are variables are extracted.

We adopt the corresponding descriptive phrase for each operand as its representation, as opposed to previous studies that use the numerical variable alone [18]. For instance, consider the sentence, "The total debt was $28.5 billion on December 31, 2015." The phrase representation used as the operand would be *VP* "was $28.5 billion on December 31, 2015." By utilizing phrase representations, we can capture the contextual information and convey the intended message more effectively, as the numerical variable lacks inherent semantic meaning. Specifically, for variables from the table, we consider the unit description sentence; for variables from plain text, we consider the constituency parsing *VP* or *PP* ancestor node of the variable, which has a *NP* sibling node.

The operator prediction task treats the operator y_{op} as a relation between the operand variables and employs relation classification methods [4,119] to construct the operator prediction module. However, unlike relation classification tasks, some program functions may have more than two operands (e.g., *table_sum* can have three or more operands). So instead of concatenating the token representations, the final representation h_{op} of the operator is calculated by averaging the PLM *Encoder* representations $h_0, ...h_m$ of the m operands' first token. A feed-forward network (FFN) and softmax function map the representation to the operator prediction. The final loss is defined as:

$$\begin{aligned} h_{op} &= avg(h_0, ...h_m) \\ \mathcal{L}_{op} &= NLL(softmax(FFN(h_{op}), y_{op}) \end{aligned} \tag{4.3}$$

Where y_{op} belongs to {*'add', 'subtract', 'multiply', 'divide', 'exp', 'greater', 'table_sum', 'table_average', 'table_max', 'table_min'*}.

4.3.3 VKM: Variable Keyphrase Masking

This task trains the model to recover masked keyphrases that describes the variables from a masked question and evidence sequence example. We leverage two keyphrase extraction methods. To effectively extract keyphrases that describe the required variables, we use cell-based evidence that contains less noise of irrelevant variables.

First, we use TextRank [74] over the concatenation of the question text and the gold evidence to extract keyphrases. TextRank constructs a graph over tokens considering co-occurrence. Then PageRank [78] algorithm is applied over the token graph to rank token importance. We only use the keyphrases that appear twice or more for masking, to ensure they are valid descriptions for the required variables.

Second, we consider a feature of table data, that headers naturally form keyphrases that describe the variables. However, not all headers could be reconstructed given the context, such as *'The acklen west end'* in Figure 4.2, the name of this community is only given in the table row. We use the headers that

appear twice or more for keyphrase masking, that these headers describe more than one variable, implying the relation of two or more variables.

We follow the Masked Language Model pretraining of BERT [24] to train the model to recover the keyphrases. Given the concatenation of question and cell-based evidence, each keyphrase is randomly masked for once occurrence. The masked input text is given to the transformer-based language model, and the model is trained to predict the token at the masked positions with Cross Entropy Loss $\mathcal{L}_{mask}$.

4.3.4 Multi-task Pretraining

To perform multi-task training, we leverage a streamlined version of MT-DNN [66, 67]. We construct fixed mini-batches $\{b_t\}$, that in each mini-batch all examples are of the same pretraining task t. In each training step, a mini-batch b_t is selected and the model is updated according to the task-specific loss $\mathcal{L}_t$ for the task t. This optimization procedure could be seen as an approximation of the sum of multi-task objectives $\mathcal{L}_{rank}$, $\mathcal{L}_{op}$ and $\mathcal{L}_{mask}$.

4.4 Experiments

4.4.1 Dataset

In this study, we conducted experiments using two datasets: the FinQA dataset [17] and the more challenging MultiHiertt (MH) dataset [116] as shown in Table 4.1. FinQA utilizes flat tables and each passage only has one table. MultiHiertt presents a higher level of complexity, which contains both span prediction and program prediction questions and features an average of 3.89 hierarchical tables per example.

To the best of our knowledge, the TAT-QA dataset [120] is the only other dataset that investigates hybrid table-and-text numerical reasoning. However, the experimental setups and textual domains of TAT-QA are highly similar to those of FinQA.

Dataset	#Q	#Train/Dev/Test	#Op. Max./Avg.	#E. Max./Avg.
FinQA	8,281	6,251/883/1,147	6/1.54	9/1.71
MH	10,440	7,830/1,044/1,566	24/1.74	20/2.74

Table 4.1: Statistics of FinQA and MultiHiertt. #Q denotes the total number of questions. #Op. denotes the number of the operator in the program solution per example. #E. denotes the number of gold evidence pieces per example.

4.4.2 Evaluation Metric

We follow Chen et al. [17] and evaluate the overall performance of FinQA dataset using two metrics: execution accuracy (exe) and program accuracy (prog). For MultiHiertt, we follow Zhao et al. [116] and report exact matching (EM) and adopted numeracy-focused F1.

To examine the performance of the retriever, we report the top-3 recall (**R@3**) and top-5 recall (**R@5**) of the gold evidence on FinQA since it only has program solution questions. The recall metric denotes the proportion of successfully retrieved gold evidence out of the total gold evidence.

4.4.3 Implementation Details

Following FinQANet, we use **BERT-Base** [24] for the retriever and give the comparison of our method and the baseline on two PLMs for the program solver: **BERT-Base** and **RoBERTa-Large** [68] to demonstrate the effectiveness of our method on different scales of PLMs and different pretraining strategies.

For the retriever, we do not apply **VOP** and **VKM**, which are designed for sub-program construction. We observe that the retriever needs to handle a large volume of irrelevant information. To imitate the massive negative samples, we add an additional irrelevant evidence to the ranking sets during pretraining, namely noisy Variable Integrity Ranking ($\mathcal{N}$**VIR**) for the retriever. For the program solver, we apply all three pretraining tasks.

For MultiHiertt, we follow Zhao et al. [117] and extend FinQANet with a question type classification module and span prediction module to solve span prediction questions. We use vanilla PLMs for these modules since they do not

Model	PLM	FinQA Dev(%)		FinQA Test(%)		MH Dev(%)		MH Test(%)	
		exe	**prog**	**exe**	**prog**	**EM**	**F1**	**EM**	**F1**
NeRd	B-B	23.83	22.56	48.57	46.76	-	-	-	-
Longformer	-	47.53	45.37	21.90	20.48	2.71	6.93	2.86	6.23
ELASTIC	R-L	-	-	62.16	57.54	-	-	-	-
FinQANet	B-B	52.10	49.83	51.43	49.26	35.69	37.81	34.32	36.17
Ours	B-B	55.58	52.32	55.70	53.55	39.65	40.40	37.35	37.74
FinQANet	R-L	63.19	61.11	61.95	59.81	37.05	39.96	36.22	38.43
Ours	R-L	**66.82**	**63.49**	**65.51**	**63.28**	**43.20**	**43.94**	**41.70**	**42.09**
Expert	-	-	-	91.16	87.49	-	-	83.12	87.03
Crowd	-	-	-	50.68	48.17	-	-	-	-

Table 4.2: Main results of our method and baselines. **Dev** denotes performance on the development set. **Test** denotes performance on the test set. **PLM** denotes the PLM used for program solver.

involve numerical reasoning.

Our method is implemented using Pytorch [80] and Transformers [108]. BERT-based models are trained on an NVIDIA 3090 GPU with 24G memory for approximately 12 hours, while Roberta-large models are trained on an NVIDIA A100 GPU with 40G memory for approximately 24 hours. For pretraining, we use a batch size of 4 and trained for 5 epochs. The initial learning rate is set to 5e-6, and we employ a warm-up proportion of 0.1.

For fine-tuning, we followed the hyperparameters of FinQANet [17], except for the learning rate. We observed that training failed to converge using the original settings on transformer-large PLMs, and thus add a cosine learning rate scheduler with 50 steps of warm-up, gradient clipping with a maximum norm of 1.0, and weight decay of 1e-5 to avoid gradient explosion. To prevent overfitting, we utilized loss-based early stopping with the patience of 30 epochs and then selected the best model among the patience period for testing using development set accuracy. We report results based on the average of three fixed random seeds. The top-3 retrieved evidence scored by the retriever is used as input for the program solver.

Model	Dev		Test	
	R@3	**R@5**	**R@3**	**R@5**
BERT-Base	90.47	93.91	88.98	93.56
VIR	90.97	94.22	89.28	**94.08**
$\mathcal{N}$VIR	**91.66**	**95.12**	**90.05**	93.66

Table 4.3: Results of retriever performance on devlopment set and test set. **R@n** denotes the top-n retriever recall of gold evidence. **VIR** denotes using Variable Integrity Ranking as pretraining task. **$\mathcal{N}$VIR** denotes using noisy Variable Integrity Ranking as pretraining task.

4.5 Results and Analysis

4.5.1 Main Results

The experiment results are shown in Table 4.2. We compare our method with various competitive methods[1]. **NeRD** [86] assigns plus, minus or zero to all variables and sums the signed variables. **Longformer** [5] uses a PLM designed for long text as the encoder and takes in all evidence text as the input. **ELASTIC** [114] uses an adapted program solver that separates the generation of operators and operands; **Crowd** and **Expert** refers to crowd workers and professional expert human performance reported in Chen et al. [17], where the latter result serves as an upper bound of systems. **FinQANet** is results of our re-implementation of the baseline model as described in section 4.2.

Our method surpasses all baselines and achieves the best performance on both PLMs. Specifically, on FinQA test set we achieve 4.37% execution accuracy and 4.29% program accuracy improvement with BERT-Base solver, and 3.56% execution accuracy and 3.47% program accuracy improvement with RoBERTa-Large solver; on MultiHiertt test set we achieve 3.03% EM and 1.57% F1 improvement with BERT-Base solver, and 4.48% EM and 3.66% F1 improvement with RoBERTa-Large solver. These results demonstrate the effectiveness of our whole

[1]As the official github implementation states, there was a data-leakage in the early version of FinQA dataset, all results reported in this chapter remove this bug and could exist discrepancies with the results in the original chapter.

program and sub-program level pretraining approach.

4.5.2 Ablation Studies

To further understand the performance of our method, we conducted ablation studies to break down the performance of each component. We focus on FinQA dataset since it only contains program solution problems and explicitly shows the improvement of numerical reasoning.

Retriever Recall In Table 4.3, we present a comparison of the retriever performance using different pretraining settings. Our proposed method outperforms the BERT-Base baseline on all evaluation metrics under both settings of Variable Integrity Ranking. Specifically, $\mathcal{N}$**VIR** achieves the best top-3 recall on both the development set and test set, gaining 1.19% points and 1.07% points of improvement, respectively. Additionally, **VIR** outperforms the baselines and achieves the best top-5 recall performance on the test set. These results demonstrate that our whole program level pretraining task is beneficial for the retriever, as it helps the model learn to distinguish irrelevant evidence. Furthermore, the noisy Variable Integrity Ranking setting further enhances the model's ability to extract evidence from massive negative irrelevant examples, although this advantage diminishes as the number of retrieved evidence increases.

Program Prediction Accuracy To investigate how the pretraining tasks benefit the final program prediction, we conducted an ablation study evaluating the overall execution accuracy and program accuracy performance. As shown in Table 4.4, our method substantially improves program prediction results. Specifically, the retriever $\mathcal{N}$**VIR** on the test set gains 0.79% execution accuracy and 0.61% program accuracy with BERT-Base as the program solver, and 1.43% execution accuracy and 1.57% program accuracy with RoBERTa-Large as the program solver. The improvement in retriever recall, resulting from the whole program level pretraining, is directly reflected in the final prediction accuracy. For the program solver, we examine the performance of using the three pretraining tasks separately, to verify their individual effectiveness. We observed that all

R	P-S	Dev(acc/%)		Test(acc/%)	
		exe	**prog**	**exe**	**prog**
BERT-Base					
–	–	52.10	49.83	51.43	49.26
$\mathcal{N}$VIR	–	53.00	50.85	52.22	49.87
$\mathcal{N}$VIR	VIR	54.81	52.22	53.96	51.22
$\mathcal{N}$VIR	VOP	53.22	51.64	52.97	51.04
$\mathcal{N}$VIR	VKM	52.10	49.60	52.57	50.13
$\mathcal{N}$VIR	ALL	**55.58**	**52.32**	**55.70**	**53.55**
RoBERTa-Large					
–	–	63.19	61.11	61.95	59.81
$\mathcal{N}$VIR	–	65.57	62.40	63.38	61.38
$\mathcal{N}$VIR	VIR	66.59	63.65	64.87	62.86
$\mathcal{N}$VIR	VOP	66.16	62.28	64.44	61.60
$\mathcal{N}$VIR	VKM	65.46	62.51	64.60	62.25
$\mathcal{N}$VIR	ALL	**66.82**	**63.49**	**65.51**	**63.28**

Table 4.4: Ablation study using different pretraining tasks. **R** denotes retriever and **P-S** denotes program solver. – denotes the PLM baselines with no additional pretraining. **VOP**/**VKM** denotes using Variable Operator Prediction/Variable Keyphrase Masking as pretraining task. **All** denotes using all three pretraining tasks.

three pretraining tasks improve the performance compared to using no auxiliary pretraining tasks. Among the three tasks, **VIR** achieves the most improvement, 1.74% execution accuracy and 1.35% program accuracy improvement on BERT-Base, and 1.49% execution accuracy and 1.48% program accuracy improvement on RoBERTa-Large. These results show that distinguishing the required and irrelevant variables is crucial for the task. Additionally, the two sub-program level tasks also improved the test set accuracy of the program solver on both PLMs, demonstrating the effectiveness of fine-grain level supervision. We achieved the best results on both PLMs applying all three tasks on the program solver.

Dataset	Task	VIR	VOP	VKM	All
FinQA	VIR	93.78	-	-	94.91
	VOP	-	87.74	-	92.20
	VKM	-	-	52.43	68.37
MH	VIR	93.62	-	-	96.88
	VOP	-	83.17	-	86.92
	VKM	-	-	58.37	71.77

Table 4.5: Results of pretraining tasks accuracy.

4.5.3 Pretraining Task Performance

To investigate the model's ability to fit the objectives of the auxiliary tasks, we examined the RoBERTa-Large encoder performance of the three pretraining tasks on the FinQA and MH development set. As shown in Table 4.5, the model is able to fit the objectives of the three tasks, achieving high accuracy of over 85% accuracy for VIR and VOP. Because long consecutive tokens are masked in VKM, the task is relatively challenging. We observed that multi-task pretraining can benefit the results of each individual task. These results demonstrate that jointly training the three tasks can further improve the model's ability to examine required variables at the whole program level and construct sub-programs at the sub-program level.

4.5.4 Case Study

As shown in Figure 4.3, we conducted a case study on FinQA dataset to further investigate how our method improves the whole program and sub-program reasoning of the model. We compared the results of the original solver and our proposal using RoBERTa-Large, given the same $\mathcal{N}$VIR retriever inputs. In case 1, while the baseline found the required variables, it failed to construct the *'divide'* sub-program. However, our method, benefiting from fine-grain supervision, successfully predicted the correct program. In case 2, although the baseline model predicted the correct equation skeleton, it did not capture the information of *'consolidated subsidiaries'* and was misled by the irrelevant *'Minority Interests'*

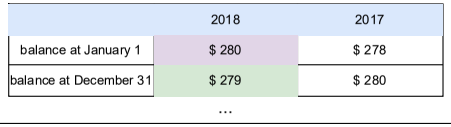
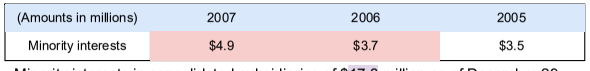
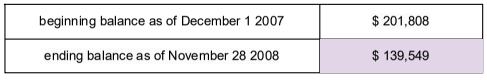
Model Input	Question	Program
	**Question:** In 2018 what was the percent of the decline in the uncertain tax positions?	**Gold Program:** divide(subtract(279, 280), 280) **Baseline Prediction:** subtract(279,280) (×) **Ours Prediction:** divide(subtract(279, 280), 280) (✓)
 Minority interests in consolidated subsidiaries of $17.3 million as of December 29, 2007, and $16.8 million as of December 30, 2006 , are included in 201cother long-term liabilities 201d on the accompanying consolidated balance sheets. ...	**Question:** What is the percentage change in the balance of the minority interests in consolidated subsidiaries from 2006 to 2007?	**Gold Program:** divide(subtract(17.3, 16.8), 16.8) **Baseline Prediction:** divide(subtract(4.9,16.8),3.7) (×) **Ours Prediction:** divide(subtract(17.3, 16.8), 16.8) (✓)
 As of November 28, 2008 , the combined amount of accrued interest and penalties related to tax positions taken on our tax returns and included in non-current income taxes payable was approximately $15.3 million. ...	**Question:** The combined amount of accrued interest and penalties related to tax positions taken on our tax returns and included in non-current income taxes payable was what percent of the total ending balance as of November 28 2008?	**Gold Program:** divide(15.3,divide(139549, const_1000)) **Baseline Prediction:** divide(15.3,139549) (×) **Ours Prediction:** divide(15.3,139549) (×)

Figure 4.3: Case study on FinQA test set. We abbreviate unused model input evidence.

evidence in the model input when selecting variables. The Variable Integrity Ranking guides the model to focus on the required variables and generate the correct solution program. In case 3, although both systems captured the correct variable and operator information, they could not perform unit conversion of *'million'* and *'percentage'* and failed the prediction. The model needs further guidance to understand numerical concepts such as unit conversion, which we consider as future work.

4.6 Related Works

4.6.1 Numerical Reasoning

Machine Reading Comprehension (MRC) style tasks predict a span or an option from multi-choice, such as DROP [26], which predicts a text or number span out of Wikipedia documents and questions and NumGLUE [75] that evaluates eight tasks that require simple arithmetic understanding. Solution prediction style tasks predict a semantic parsing style program such as MathQA [1], a mathematical equation such as Math23K [104] and MAWPs [52], or a rational textual description of the reasoning process [21]. For tabular numerical reasoning, HiTab [19] dataset considers numerical reasoning question and program pairs on hierarchical tables.

For Table-and-Text Hybrid numerical reasoning, FinQA [17], TAT-QA [120] and MultiHierr [116] consider numerical reasoning on financial reports. All questions in FinQA require program solution prediction, while a proportion of questions in TAT-QA and MultiHierr require span prediction.

4.6.2 Pretraining for Question Answering

Various post-pretraining methods have been proposed for question answering. Span prediction tasks are utilized to benefit MRC [23, 33, 45, 85], which also use token masking recovery. They aim to use cloze-like data to learn MRC, while our approach focuses on fine-grain level sub-tree construction. To enhance numerical reasoning skills of MRC systems, Geva et al. [32] and Pi et al. [82] pretrains the model to generate code programs and equations. Feng et al [28] maps entity representations with numbers representations for Knowledge-Graph question

answering.

The closest studies to our method are MWP-BERT [61] and FORTAP [18] that consider solution prediction style numerical reasoning. Both studies leverage a series of pretraining tasks to understand value magnitude and predict components of the solution program, and the most similar task to our proposal is operator prediction. However, their studies limit to considering only the variable number for pretraining, while our proposal considers the contextual information of operands.

4.7 Summary of This Chapter

In this chapter, we propose three solution program centric auxiliary pretraining tasks at both the whole program level and sub-program level. At the whole-program level, we propose the Variable Integrity Ranking pretraining task, which guides the model to distinguish required and irrelevant variables in the noisy input. To further enhance the model's ability to learn the underlying reasoning process, we propose two additional pretraining tasks: Variable Operator Prediction and Variable Keyphrase Masking. These tasks help the model perform accurate sub-program construction. Our experimental results demonstrate the effectiveness of our method, showcasing substantial improvement on two datasets, namely FinQA and MultiHiertt. The improvements are observed across PLMs of different scales. These results highlight the potential of leveraging program annotations and underscore the need for further exploration of various pretraining methods in future studies.

Chapter 5

Textual Enhanced Contrastive Learning for Solving Math Word Problems

5.1 Introduction

The challenges encountered during the training process, such as noise introduced by exposure bias in sequence-to-sequence (seq2seq) models and the mismatch between model predictions and annotations, highlight the complexity of solving math word problems (MWPs). It is evident that addressing these challenges requires a deeper understanding of the contextual information rather than relying solely on shallow keyword matching. For example, as shown in Figure 5.1, while the first problem shares high token-level overlapping with the third problem, the underlying mathematical logic is different. While on the other hand, the first and second problems have very low similarity at the textual level, while the equation solution is the same. The challenge of the task is that the underlying mathematical logic would change even with minor modifications in the text. While neural network based models have greatly boosted the performance on benchmarks datasets, Patel et al. [81] argued that state-of-the-art (SOTA) models use shallow heuristics to solve a majority of word problems, and struggle to solve challenge sets that

Problem: $P = (T, E)$

T : Dave bought n_0 boxes of chocolate candy and gave n_1 to his little brother. If each box has n_2 pieces inside it, how many pieces did Dave still have?

E : $(n_0 - n_1) * n_2$

Different Textual, Similar logic: $P^+ = (T^+, E^+)$

T^+ : A new building needed n_0 windows. The builder had already installed n_1 of them. if it takes n_2 hours to install each window how long will it take him to install the rest?

E^+ : $(n_0 - n_1) * n_2$

Similar Textual, Different Logic: $P^- = (T^-, E^-)$

T^- : For halloween Faye got n_0 pieces of candy. she ate n_1 pieces the first night and then her sister gave her n_2 more pieces. How many pieces of candy does Faye have now?

E^- : $n_0 - n_1 + n_2$

Figure 5.1: Example of positive data point $P^+ = (T^+, E^+)$ and negative data point $P^- = (T^-, E^-)$ for an anchor $P = (T, E)$.

have only small textual variations between examples.

Motivated by recent progress in contrastive learning methods, which is a flexible framework that has been successfully employed to representation learning in various fields [20, 27, 31], we propose Textual Enhanced Contrastive Learning, which is an end-to-end framework that uses both textual and mathematical logic information to build effective representations. For each *anchor* data point, we find the *hard example* triplet pair, which consists of a textual-different but logic-similar *positive* data point P^+, and a textual-similar but logic-different *negative* data point P^-. Our method aims to learn an embedding space where the vector representations of P and P^+ in Figure 5.1 are mapped close together, since they hold the same mathematical logic even though the textual expression is entirely different; on the other hand, because P and P^- have similar textual expression but different mathematical logic, their vector representations could be separated

apart.

While previous studies also used Contrastive Learning to improve representations for solving MWPs [59], their method is limited to supervised contrastive learning, ignores textual information during constructing the contrastive learning pairs and requires two-step pre-training and re-training. Our method pushes the model to learn better text representations and understand the most minor textual variance from these textual enhanced *hard samples* from both *supervised* and *self-supervised* perspectives. Self-supervised contrastive learning encourages the model to learn relevant features by constructing contrasting views of the same data, promoting a deeper understanding of the underlying data distribution. This leads to improved generalization and performance on downstream tasks. Meanwhile, self-supervised contrastive learning is more robust to noisy or erroneous labels, as it inherently learns to distinguish relevant information from irrelevant noise. Overall, these advantages position self-supervised contrastive learning as a powerful and promising paradigm for representation learning in the absence of traditional supervision.

To build such triplet pairs, we use a retrieval-based method to search in the training data. We consider the equation annotation as the representation of the mathematical logic in the example, and retrieve a positive and negative bag of data points according to equation similarity. Then we further use textual similarity to choose the *hard examples* in the bags, where positive examples have low textual similarity with the anchor and vice versa. Given such *hard sample* data, Contrastive Learning could empower the representations by leading the model to distinguish these potential disorienting examples in the training stage.

Such approaches to retrieving triplet pairs from human-annotated training data via label annotation are considered as *supervised* contrastive learning. Another research line of contrastive learning is *self-supervised* contrastive learning, which does not require labeled data and use data augmentation methods to generate the positive or negative data points [15, 36, 39]. In the task of solving MWPs, we can leverage *self-supervised* supervision by generating new examples via performing synchronized changes to text and equations. The generated data is naturally *hard sample* data, because the textual expression is similar to the origin

example, while the equation could be either changed or the same. Specifically, we leverage Reversed Operation-based Data Augmentation [64] and a Question Reordering-based augmentation to form new data points. By enhancing the model to detect the small perturbations in the augmented examples, contrastive learning forces the model to learn more effective representations of contextual information.

We conduct experiments on two widely used datasets, the English dataset ASdiv-A [73] and the Chinese dataset Math23K [104]. To further investigate how our method improves the ability of the model to detect small textual perturbations, we collect a Chinese challenge set Hard Example (HE)-MWP. We perform experiments on two challenge sets of MWPs, the English Asdiv-Adv-SP dataset [55] and the Chinese HE-MWP dataset. Experimental results show that our method achieves consistent gains under different languages and settings, demonstrating the effectiveness of our method.

5.2 Related Work

5.2.1 Contrastive Learning

Contrastive Learning was first adopted in Computer Vision to learn representations of images via self-supervision without human annotation [15, 36, 39]. Self-supervised contrastive learning is applied in NLP to learn sentence representations. Back translation [27] and dropout [31] are used to construct positive-negative contrastive learning triplets. These perturbation-based techniques are not suitable for MWP solvers, that MWPs are sensitive to small textual variance and the perturbation might introduce noise.

[48] first introduced supervised contrastive learning in Computer Vision by modifying the loss to allow supervision from label annotations. In NLP, various studies have introduced natural language inference (NLI) datasets as supervised annotations for contrastive learning [31, 88]. The agreement of equation annotations of MWPs can be considered a form of NLI, that our supervised contrastive learning could be considered a transformation of these methods.

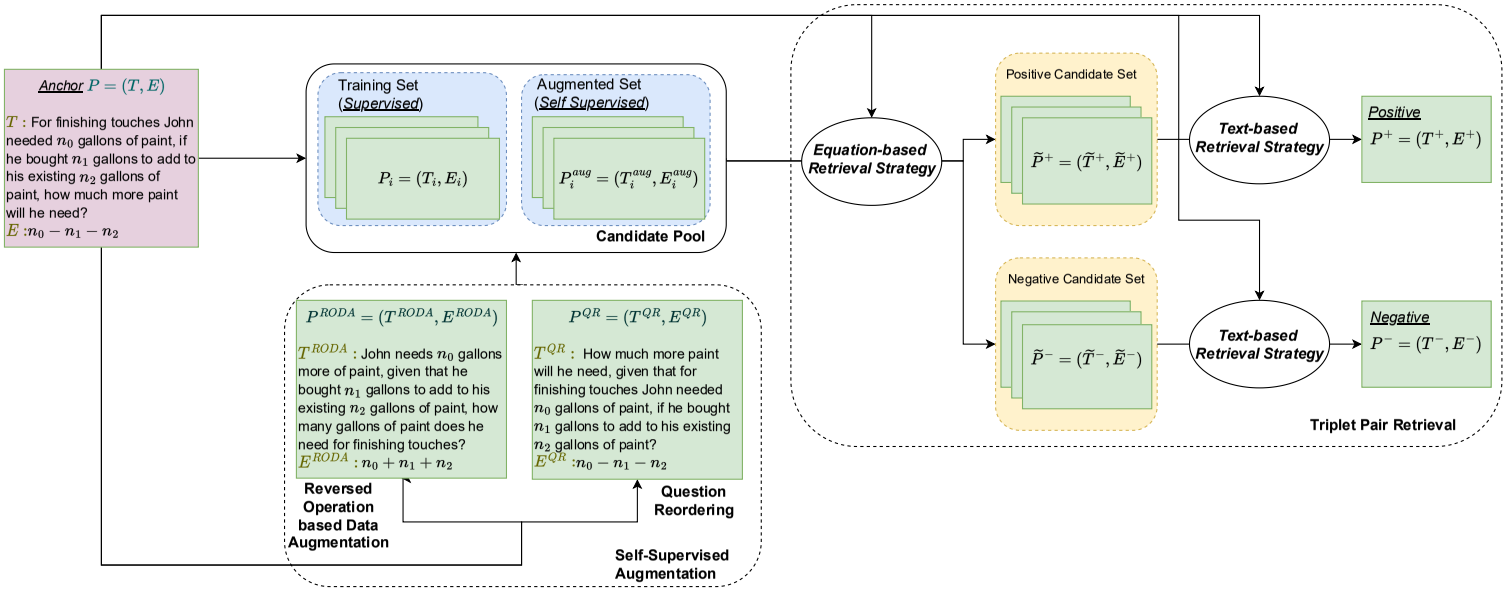


Figure 5.2: Overview of the contrastive learning triplet pairs retrieval procedure.

5.3 Methodology

We use Contrastive Learning to obtain text features with high differentiation of small perturbations, so that for each *anchor* data point $P = (T, E)$, where T stands for the text and E stands for the equation, we construct a pair of examples *positive* data point $P^+ = (T^+, E^+)$ and *negative* data point $P^- = (T^-, E^-)$, and then use contrastive learning loss to map the representation of P and P^+ closer and vice versa. The pipeline of the triplet pairs retrieval is shown in Figure 5.2. We first construct a candidate pool, which consists of *supervised* training data $\{P_i\}$ and augmented *self-supervised* data $\{P_i^{aug}\}$ as shown in the blue part of Figure 5.2. The *self-supervised* data is generated by two methods, Reversed Operation based Data Augmentation (RODA) and Question Reordering (QR), which is explained in Section 5.3.1. Then we perform two-step retrieval to retrieve the triplet pairs as described in Section 5.3.2. We first use an equation-based retrieval strategy to extract *positive* candidate set $\{\widetilde{P}^+\}$ and *negative* candidate set $\{\widetilde{P}^-\}$, and then further introduce textual information by choosing one example from the candidate set via a text-based retrieval strategy. Finally, we train the MWP solving model that maps T to E by considering both the contrastive learning and solution equation generation objective, as described in Section 5.3.3.

5.3.1 Enriching Candidate Pool via Self-Supervised Augmentation

The *self-supervised* examples are challenging for the model to distinguish; while the perturbation in the text expression is extremely subtle, the corresponding mathematical logic could still change. Compared to the supervised examples that are retrieved from the training data, these *self-supervised* samples place a higher demand on the model's ability to detect subtle changes and understand contextual information. We generate task-orientated augmented examples from training set data point $P = (T, E)$ via two methods that obtain reliable new text-equation examples by modifying the text and equation in the same logic at the same time. We split the sentences with punctuation marks to a question followed by various declarative sentences $T = \{S_1, S_2, ..., S_{k-1}, Q_k\}$. The question sentence is always

the last sentence for Asdiv, and we check whether interrogative pronouns are in the last sentence for Math23K.

Question Reordering

We move the question to the front of the MWP to form a reordered new MWP similar to [55]. Given a problem text $T = \{S_1, S_2, ..., S_{k-1}, Q_k\}$, we move the question Q_k to the front of the problem text to form a new problem text $T^{QR} = \{Q_k, S_1, ..., S_{k-1}\}$ while the rest of the text remains the same. We simultaneously edit the equation E^{QR} so that the variables match with the new text order. The new example $P^{QR} = (T^{QR}, E^{QR})$ could either be a positive example that holds the same equation as P or a negative example that holds a different equation since the variable order might change during the reordering. The high textual similarity but rotated variable order pushes the model to learn representations that can differ from these small textual perturbations.

Reversed Operation based Data Augmentation

We perform RODA [64] that generates a new example by asking a question about one of the original given variable. Given a problem text $T = \{S_1, S_2, ..., S_{k-1}, Q_k\}$ where the question Q asks about an unknown variable n_{ans}, RODA chooses a known variable n in one of the declarative sentence S_i, and then generates a problem text which asks about this variable. To generate such an example, S_i is transformed to a question Q_{S_i} which asks a question of n, while Q is transformed to a declarative sentence S_k describing n_{ans}. We reorder the problem text by swapping the two sentences, that a new problem text $T^{RODA} = \{S_1, ...S_k, ..S_{k-1}, Q_i\}$ is generated. Simultaneously we edit the equation by resolving the equation expression E^{RODA} of n given n_{ans}. While $P^{RODA} = (T^{RODA}, E^{RODA})$ has a very similar textual description of P, the underlying equation could be completely different, which could benefit the model via contrastive learning. RODA requires text parsing and transformation rules to modify the text and equation. For Chinese, it can cover 93% of the examples, and for English, it covers 60% of the examples. The generated text has a 0.83 out of 1 coherent score reported by human evaluation by [64].

5.3.2 Triplet Pair Retrieval

This subsection is majorly the proposal of Yibin Shen, the co-first author of this paper, we leave this subsection in this thesis to describe the complete methodology. We construct the positive and negative triplet pairs from both textual and logical perspectives. For a given problem P, the positive sample P^+ is considered to be a problem with similar equation expressions but relatively different text descriptions; the negative sample P^- is considered to be a problem with highly textual similarity but different equation expression. However, it requires a time-consuming bruce-forth enumeration of all possible example pairs to find such optimal positive and negative samples. Considering the computational complexity, we break down the retrieval to a two-step pipeline. we adopt a heuristic searching algorithm to construct positive and negative samples (P^+, P^-) as follows:

1. Construct a similarity matrix M of all equation expressions $\{E_1, E_2, ...E_n\}$ in the training set, where M_{ij} is the similarity of equation expression E_i, E_j.

2. For a given anchor P, Retrieve a *positive* candidate set $\{\widetilde{P}^+\}$ and a *negative* candidate set $\{\widetilde{P}^-\}$ of samples from the training set of the data via equation expression similarity.

3. Extract the best *positive* example P^+ and the best *negative* example P^- via textual similarity.

We investigate various strategies to retrieve (P^+, P^-) from both equation-based and text-based perspectives.

Equation-based Retrieval Strategy

To evaluate the equation similarity during the retrieval, we design an equation similarity metric Sim_{eq} based on length-wise normalized tree edit distance (TED). TED is defined as the minimum-cost sequence of node operations that transform one tree into another and is a well-known distance measure for hierarchical data.

We define the TED of two equation expressions E_1, E_2 as the TED of their abstract syntax tree. The similarity of two equation expressions E_1, E_2 is defined as:

$$Sim_{eq}(E_1, E_2) = 1 - \frac{TED(E_1, E_2)}{|E_1| + |E_2|}$$

Given this equation similarity metric, we design two retrieval strategies.

Exact Match The *positive* candidate set $\{\widetilde{P}^+\}$ is constructed of the examples that meets $Sim_{eq}(E, E_i) = 1$, which means their equation expression satisfies $E = E_j$. If only the *anchor* itself holds this equation expression, the *positive* candidate set $\{\widetilde{P}^+\}$ has only the *anchor* P. The *negative* candidate set $\{\widetilde{P}^-\}$ is constructed of the examples that meets $argmax_{E_i \neq E}(Sim(E, E_i))$, which holds the closest equation considering the *anchor*.

Nearest Neighbour The *positive* candidate set is constructed of the examples that meets $argmax_{E_i, T_i \neq T}(Sim_{eq}(E, E_i))$. If no other example holds the same equation expression as the *anchor*, the *positive* candidate set $\{\widetilde{P}^+\}$ takes the examples that has the nearest neighbour equation expression. The *negative* candidate set $\{\widetilde{P}^-\}$ is constructed of the examples that meets $argmax_{E_i \neq E^+}(Sim_{eq}(E, E_i))$, which holds the closest equation considering the *positive* example.

The *positive* and *negative* candidate sets are then further screened by the text-based strategy.

Text-based Retrieval Strategy

To lead the model to differentiate mathematical logic from similar textual expressions, we use textual-based information to select the (P^+, P^-) pair. We select the **lowest** textual similarity score example from the positive candidate set $\{\widetilde{P}^+\}$, which is the example with different textual expression but the same mathematical logic; and select the **highest** textual similarity score example from the negative candidate set $\{\widetilde{P}^-\}$, which is the example with similar textual expression but different mathematical logic. We design two similarity measurement metrics for this stage.

Dataset	Math23K	Asdiv-A	HE-MWP	Adv-Asdiv
Language	zh	en	zh	en
Domain	general	general	challenge	challenge
#Train	21,162	1,218	-	-
#Dev/#Test	1,000 / 1,000	- / -	- / 400	- / 239
#Equation Templates	3,104	66	231	66

Table 5.1: Statistics and details of the datasets.

BERTSim Sentence-BERT (SBERT) is a strong sentence representation baseline model [88]. We calculate the cosine similarity of the SentBERT representation of the two sentences to obtain the similarity score:

$$Sim_{text}^{BERTSim} = \frac{SBERT(T_1) \cdot SBERT(T_2)}{||SBERT(T_1)||||SBERT(T_2)||}$$

The value range of $Sim_{text}^{BERTSim}$ is from $[-1, 1]$.

Bi-direction BLEU BLEU is a widely used evaluation metric for text generation that measures the similarity between the generated text and the reference. We design a Bi-direction BLEU since BLEU is a not symmetrical similarity metric, which is defined as:

$$Sim_{text}^{BiBLEU} = \frac{BLEU(T_1, T_2) + BLEU(T_2, T_1)}{2}$$

The value range of Sim_{text}^{BiBLEU} is from $[0, 1]$.

5.3.3 Training Procedure

We show the training procedure in Figure 5.3. The training loss consists of the MWP solving loss $\mathcal{L}_{solver}$ and the contrastive learning loss $\mathcal{L}_{cl}$.

MWP Solving Model We follow [59] and use the strong baseline model BERT-GTS as MWP solving model. The pre-trained language model BERT, which provides strong textual representations, is leveraged as the encoder. For the decoder,

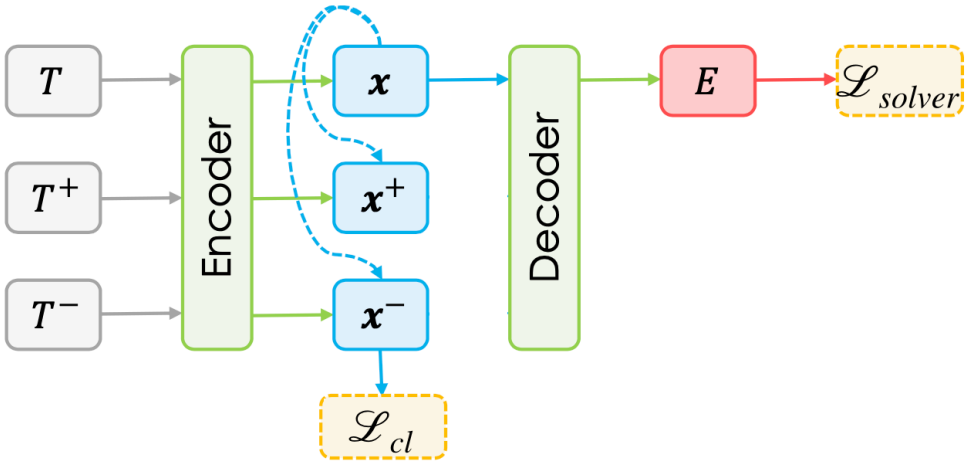


Figure 5.3: Overview of the training procedure.

we use Goal-driven tree-structured MWP solver (GTS) [109]. GTS directly generates the prefix notation of the solution equation by using a recursive neural network to encode subtrees based on the representations of its children nodes with the gate mechanism. With the subtree representations, this model can well structured information of the generated part to predict a new token.

Contrastive learning Contrastive learning is performed on triplets pairs (P, P^+, P^-) by pulling the representations of T and T^+ together and pushing apart the representations of T and T^-. We follow the contrastive learning framework in [15], which takes an in-batch cross-entropy objective. Let x_i denote the encoder representation of P, the training objective for (x_i, x_i^+, x_i^-) within the batch of N triplet pairs is:

$$\mathcal{L}_{cl} = -\frac{1}{N}\sum_{i=1}^{N} log \frac{e^{cos(x_i, x_i^+)/\tau}}{\sum_{j=1}^{N}(e^{cos(x_i, x_j^+)/\tau} + e^{cos(x_i, x_j^-)/\tau})}$$

where τ is the temperature hyperparameter.

Model	Cand. Pool	Math23K	Asdiv-A	HE-MWP	Adv-SP
GTS [109]	-	75.6	68.5	-	21.2
G2T [115]	-	77.4	71.0	-	23.8
pattern [59]	train	83.2	-	-	-
BERT-GTS	-	82.9	73.4	55.5	59.9
w/ CL	train	84.1	74.2	57.2	63.7
w/ RODA	RODA+train	84.3	74.3	64.1	64.1
w/ QR	QR+train	84.2	74.4	62.5	66.2
w/ All	All	85.0	74.6	69.5	66.9

Table 5.2: Results on MWP datasets. All experiments only compute MWP solving loss on the training set. The candidate pool only affects the choice of positive and negative examples in the CL loss.

Assume the prediction target equation of P is y, the final training objective is to minimize the sum of the MWP solution equation generation negative log-likelihood loss $\mathcal{L}_{solver}$ and the contrastive learning loss $\mathcal{L}_{cl}$:

$$\mathcal{L} = \mathcal{L}_{solver} + \alpha * \mathcal{L}_{cl}$$

5.4 Experiments

5.4.1 Datasets

We perform experiments on four datasets, including two widely used datasets to verify the generalization ability of our method and two challenge test sets to show further how our method can enhance the robustness of the model. We show detailed statistics of the datasets in Table 5.1.

Math23K is a Chinese dataset that contains 23,161 math word problems of elementary school level [104]. We use the standard train-test split setting of this dataset for the experiment.

Asdiv-A is the arithmetic subset of ASDiv which has 1,218 MWPs mostly up to grade level 4 [73]. Experiments of this dataset are evaluated by 5-cross validation.

HE-MWP Since no challenge dataset has been developed for Chinese MWP solving, and existing challenge datasets have limited types of equation templates, we use RODA and QR on Math23K validation set to generate examples that are semantically similar to the original input but deceive the model into generating an incorrect prediction. We randomly sample a subset of 600 examples from the RODA result of the development set of Math23K and then manually delete the examples that the text is not coherent. Then we randomly select 400 examples out of this cleaned subset.

Adv-SP is challenge set of Asdiv-A, which is constructed of adversarial examples [55]. These adversarial examples are generated by sentence paraphrasing.

Results of the challenge datasets are tested on the highest performance models trained on the corresponding benchmark datasets.

There exists other MWP datasets, which are relatively less challenging such as ALG514, DRAW1K and MAWPS [53, 56, 101], or noisy such as Dolphin18K [44] or use semantic parsing as annotation such as MathQA [1]. We use the two benchmarks Math23K and Asdiv-A because they are both clean and challenging with mathematical equation annotations.

5.4.2 Implementation Details

We use two language-specific BERT-base models as the problem encoder[1]. For both models, the maximum text length of the encoder is fixed at 256, and the maximum equation generation length of the decoder is fixed at 45. The decoder embedding size is 128. The batch size is 16, with learning rate of 5e-5. We tune the hyperparameters temperature τ in the set of {0.05, 0.1, 0.2} and α in the range [0.1, 0.9]. Experiments of the Chinese datasets are conducted on V100 and

[1]English: https://huggingface.co/bert-base-uncased, Chinese: https://huggingface.co/yechen/bert-base-chinese

	Eq Strategies	
Text Strategies	EM	NN
Random	83.2	82.3
BERTSim	83.6	83.1
Bi-BLEU	**84.1**	83.2
BERT-GTS	82.9	

Table 5.3: Results of different retrieval strategies for supervised contrastive learning. EM denotes exact match. NN denotes nearest neighbour. Random denotes randomly choosing an example from the candidate set. BERTSim and Bi-BLEU denotes choosing the examples by similarity metric.

RTX 3090 with approximately 6 hours of runtime. Experiments of the English datasets are conducted on 1080Ti with approximately 1-hour runtime.

5.5 Results and Analysis

5.5.1 Pre-examination on Retrieval Strategy

We conduct a breakdown analysis on the most complex dataset Math23K of different retrieval strategies. We investigate the performance of different retrieval strategies for supervised contrastive learning. As shown in Table 5.3, for the equation-based retrieval strategy, the exact match equation strategy is more effective than the nearest neighbour strategy. This shows that the positive sample for the anchor must have accurate same mathematical logic for contrastive learning to benefit the performance. Both text-based retrieval strategies can improve the MWP solving performance compared to the random choosing baseline, demonstrating the effectiveness of introducing textual information for contrastive training. With textual-based retrieval, the extracted positive and negative examples would form *hard examples* that can push the model to differ textual-similar but logic different examples. Bi-BLEU also has a slightly higher performance than BERTSim. In the following experiments, we use the best combination of EM and Bi-BLEU as retrieval strategies.

5.5.2 Main Results

We show the results of our method compared with other baselines in Table 5.2. In addition to our baseline BERT-GTS model, we also investigate three strong baseline models. **GTS** [109] uses an LSTM encoder and the same decoder as BERT-GTS that generates the abstract syntax trees through a tree structure decoder in a goal-driven manner. **G2T** [115] is a graph-to-tree model that uses a graph-based encoder for representing the relationships and order information among the quantities. **Pattern** [59] proposes a pattern-based contrastive learning, that considers the equation similarity with supervised contrastive learning. We can see from the results that our method outperforms previous studies in all datasets. Compared to **Pattern CL** which ignores textual information, our method allows the model to have a stronger ability to bridge text descriptions to mathematical logic even using the same candidate pool. The *self-supervised* methods outperform the *supervised* settings, especially on challenge datasets, demonstrating the effectiveness of leading the model to learn contextual representations of small textual perturbations.

On benchmark datasets, we achieve 2.1% points of improvement on Math23K and 1.2% points of improvement on Asdiv-A. One major reason is that RODA can only generate 394 examples for the English dataset Asdiv-A. In contrast, it can generate 47,318 examples for the Chinese dataset Math23K because English has more strict grammar than Chinese. On challenge datasets, we achieve 14% points of improvement on HE-MWP dataset and 7.0% points of improvement on Adv-Asdiv-SP dataset. For HE-MWP ablation, RODA is more effective since it could introduce new mathematical logic examples. For Adv-Asdiv-SP, since QR is similar to paraphrasing techniques, it gains more improvement with self-supervised supervision.

5.5.3 Visualization and Case Study

We show T-SNE visualization results of the representations of examples from the top-five frequent equation templates in Math23K: $n_1 * n_2 / n_3$, $n_1 * n_2$, n_1 / n_2, n_2 / n_1 and $n_1 * (1 - n_2)$, which refers to orange, red, blue, green and purple in Figure

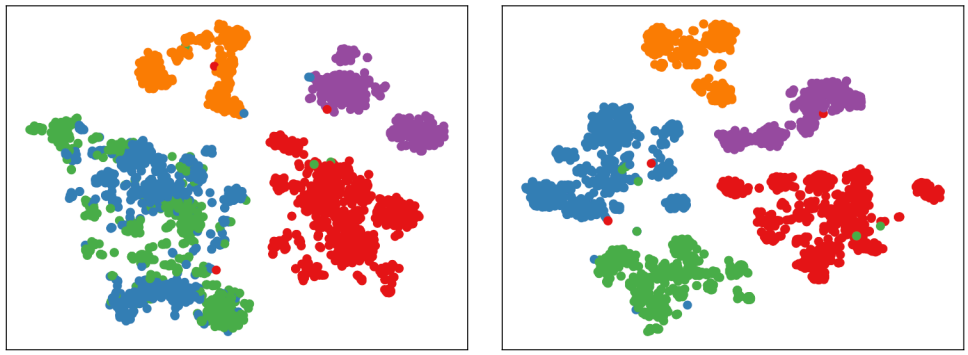


Figure 5.4: T-SNE Visualization results of BERT-GTS w/o (left) and w/ CL (right).

5.4. We can see that compared to the BERT-GTS baseline on the left subfigure, in the right subfigure, the text representations of the same equations are pulled closer via our contrastive learning, and the representation of different equations are separated apart, which shows that our method can benefit the representation learning.

We further investigate how our method improves the representation via case study. In Table 5.4 the BERT-GTS baseline could not infer from the textual description that the side area of a cylinder is the area of a rectangle but rather uses shallow heuristics when the word "cylinder" is encountered and generates the constant π. By constructing positive and negative sample pairs from both expressions and textual descriptions and changing the representation space via contrastive learning, the model is not misled by the keywords and correctly infers that the mathematical logic is to calculate the area of a rectangle so that the model with contrastive learning generated the correct result. We also show T-SNE visualization of the representation in Figure 5.5. The red dots are examples with the keyword rectangle and hold the equation $n_1 * n_2$. The blue dots are the examples that hold the equation $\pi * n_1$ or $\pi * n_2$. The green dot is the studied case. We can see that while BERT-GTS fails to separate the representation of the case from the cylinder or circle-related equations, contrastive learning helps the model to differentiate such confusing examples, learn better representations,

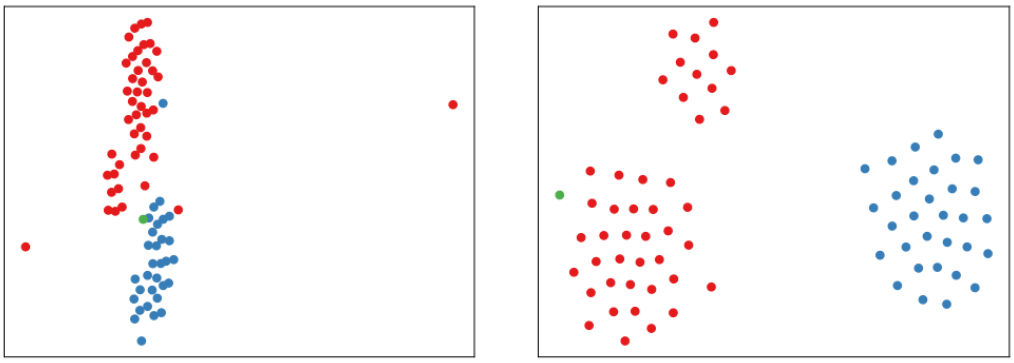


Figure 5.5: T-SNE visualization for the case study on BERT-GTS w/o (left) and w/ CL (right).

and predict the answer correctly.

5.5.4 Combination with Data Augmentation

While the high-quality and challenging augmented examples have shown remarkable effectiveness for contrastive learning, a question remains whether contrastive learning is still effective when these augmented examples are directly used as training data. Thus, we further investigate using the augmented examples as *anchors*. We use the augmented examples and the original data as training data and perform supervised contrastive learning in the training data. As shown in Table 5.5, we can see that while the augmented examples improve the performance, contrastive learning can further boost the performance, achieving SOTA results on Math23K.

5.6 Summary of This Chapter

In this chapter, we propose a Textual Enhanced Contrastive Learning framework, which leverages both *supervised* and *self-supervised* supervision to help the model understand contextual information and bridge subtle textual variance to mathematical logic. We use two novel task-specific data augmentation methods to enrich the candidate pool with examples with minor textual variance for contrastive

Text	用一张长n_1 厘米，宽n_2 厘米的长方形 纸围成一个最大 的圆柱，圆柱的侧面 积为多少平方厘米?
EN	Given a piece of paper n_1 centimeters long and n_2 centimeters wide, How many square centimeters is the lateral area of the largest cylinder enclosed by the rectangle?
w/o CL	$\pi * n_2$ (X)
w/ CL	$n_1 * n_2$ (Y)

Table 5.4: Case study on Math23K example. w/o CL denotes the BERT-GTS baseline. w/ CL denotes using contrastive learning.

Model	Acc
baseline	82.9
+QR aug w/o CL	84.9
+QR aug w/ CL	**85.2**
+RODA aug w/o CL	84.8
+RODA aug w/ CL	**86.4**

Table 5.5: Results of using augmented example for both training and contrastive learning.

learning triplet pair retrieval. We design a two-stage retrieval method to find *hard example* triplet pairs with both equation and textual information and investigate various retrieval strategies. Experimental results show that our method gained improvement on both benchmark datasets and challenge datasets in English and Chinese. We also conduct visualization for representation distribution on different equations and also on a case study, which shows our method can benefit the representation learning. With the combination of data augmentation, our method still improves the performance and achieves SOTA results on Math23K dataset.

Chapter 6

Seeking Diverse Reasoning Logic: Controlled Equation Expression Generation for Solving Math Word Problems

6.1 Introduction

The introduction of exposure bias during the training process can cause noise in sequence-to-sequence (seq2seq) models, leading to a mismatch between model predictions and annotations. This mismatch hinders the learning process and poses challenges for the model's convergence. Espcially for recent studies researchers that tackle more challenging variations of math word problems (MWPs), such as MWPs with multiple unknowns. [11, 83, 84, 101]. For human students in practice, they intuitively use diverse reasoning logic to solve MWPs. Students could consider the MWP solution from different aspects by considering diverse equivalence relations in the MWP. As we show in the upper of Figure 6.1, we can solve this problem in at least two different reasoning logic: As shown on the left side, the equation set is formed by the first reasoning logic of "considering the equivalence relation of the two sums of the cheeseburger and pizza calories given in the

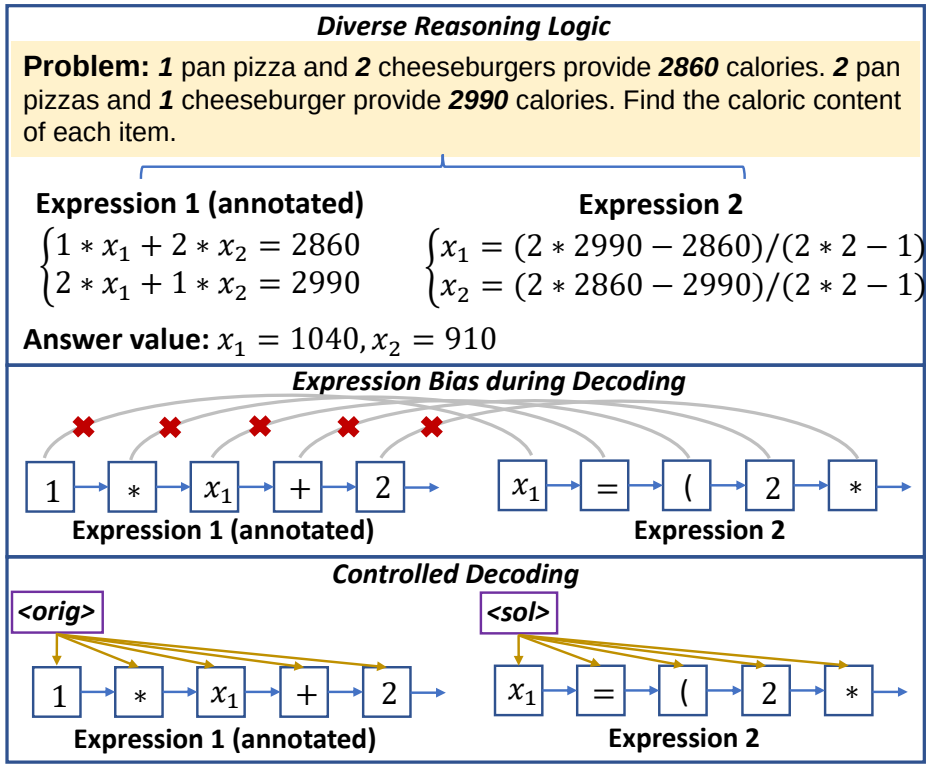


Figure 6.1: Example of diverse reasoning logic, expression bias, and our controlled expression generation. *¡orig¿* and *¡sol¿* are the pre-defined control codes.

question"; or as shown in the right side, we can follow a second reasoning logic "considering first only the equivalence relation of caloric content of the cheeseburger by offsetting the calories from the pizza". Such diverse reasoning logic could lead to diverse equation expressions, that the solution equation is written in various mathematically equivalent forms, such as expression 1 and expression 2 in the example. However, previous studies share a long-lasting limitation that they force the solver to decode a fixed equation expression supervised by human annotation. The fixed equation expression supervision used in previous studies ignores diverse mathematical reasoning, which is especially common for human students in multiple-unknown problems and complex single-unknown problems.

Meanwhile, directly introducing diverse equation expressions to the seq2seq framework in a data augmentation manner could further aggravate the issue of

expression bias, which refers to the discrepancy between the annotated equation expression and the model's correct prediction expression. As shown in the middle of Figure 6.1, even if the model makes the correct prediction of the problem, the training loss accumulated by diverse expressions could be enormous. [102] propose an equation normalization that reorders the variables in the equations as close as possible to their order in the input text. While their method could reduce the expression bias issue, they ignore the inherent diverse mathematical reasoning and limits to considering single-unknown problems.

Enlightened by recent methods in controlled text generation, which uses a control code to influence the style and topic of subsequent generated text [47], we propose a new training paradigm, where a control code guides the decoding process to consider one type of mathematical reasoning logic and decode the corresponding equation expression. As shown in the bottom Figure 6.1, the *¡sol¿* control code guides the model to consider the direct solution of each individual unknown x_1 and x_2. Not only can it reduce the expression bias problem since the control code can provide guidance for the reasoning logic, but also training on the diverse equation expressions guided by the control codes can lead to better interpretation of the MWPs by considering diverse reasoning logic. We design various control codes for both single-unknown and multiple-unknown settings to allow the model to understand different reasoning orders. We conduct experiments on a single-unknown benchmark Math23K and two multiple-unknown benchmarks DRAW1K and HMWP. Experimental results show that our method improves the performance of both settings, with a more significant improvement in the challenging multiple-unknown setting.

6.2 Methodology

This section describes the joint work with Yibin Shen. Specifically, he proposed the diverse expression augmentation, while we designed the control decoding strategy together. For each math word problem holding an original equation set $(e_1, e_2, ...)$, we generate new equation expressions based on five types of diverse mathematical reasoning logic considering the ordering logic of given variables $\{n_i\}$

and unknown variables $\{x_j\}$. i and j denote the ordered indices that the variables appear in the text. We then assign a corresponding control code to the equation expressions. The MWP solving model takes in the text and control code, and then is trained to predict the corresponding equation expression.

6.2.1 Control Codes

We consider the diverse mathematical reasoning logic in two aspects. The first aspect considers diverse reasoning orders of given variables, which reflects in the diverse expressions of the commutative law and solution form. For example, $n_1 * x_1 = n_2$ could be transformed to the solution form $x_1 = n_2/n_1$ which does not effect the mathematical equivalency. This approach is valid for both multi-unknown and single-unknown problems. The second aspect considers diverse reasoning orders of unknown variables, which reflects in the diverse expressions of equivalent equation sets. For example, swapping the equation order in the equation set does not affect the mathematical equivalency. This approach is valid for multi-unknown problems.

We preprocess the equation annotations with Sympy [72] so that they follow a predefined order similar to Wang et al. [102]. Then we generate different types of equation expressions based on these preprocessed equations.

For the first aspect, we consider three types of diverse equation expressions.

- **Commutative Law of Addition *add*** We traverse the equation in prefix order, and swap the left and right subtrees of the addition operators. For example, $x_1 = n_1 + n_2 + n_3$ would be swapped two times. We first swap the two subtrees n_1 and n_2 of the first addition operator to $x_1 = n_2 + n_1 + n_3$, and then swap the two subtrees $n_2 + n_1$ and n_3 of the second operator to $x_1 = n_3 + n_2 + n_1$.

- **Commutative Law of Multiplication *mul*** Similarly, we traverse the equation in prefix order, and swap the left and right subtrees of the multiplication operators. For example, from $x_1 = n_1 * n_2 * n_3$ to $x_1 = n_3 * n_2 * n_1$.

- **Solution Form *sol*** We consider a mathematical reasoning method that

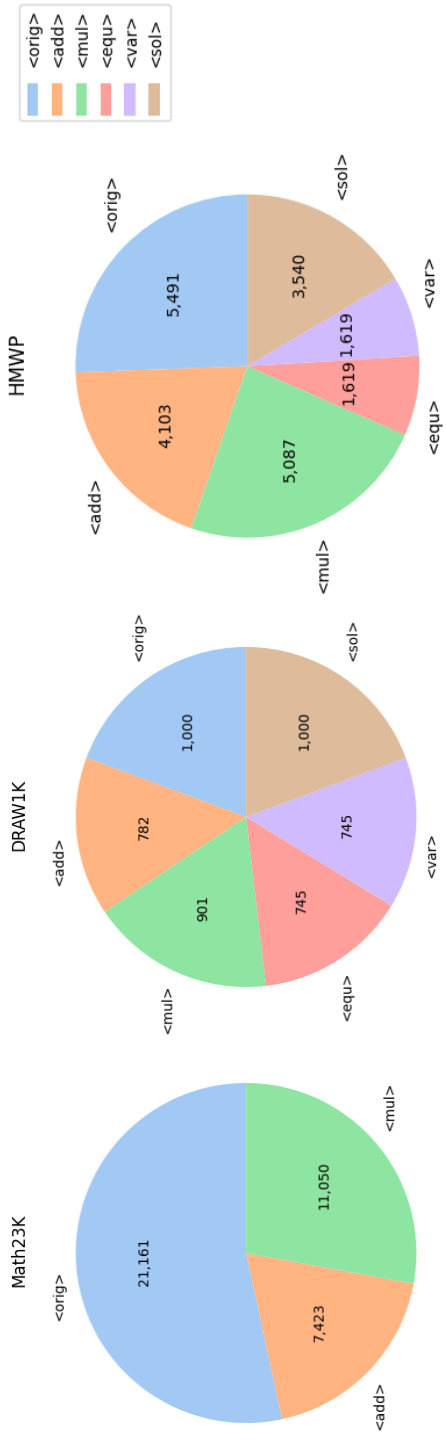


Figure 6.2: Statistics of datasets and the usage of control codes.

directly considers the solution of each unknown variable. For example, from $n_1/x_1 = n_2$ to $x_1 = n_1/n_2$.

For the second aspect, we consider two types of diverse equation expressions.

- **Equation Swapping *equ*** We swap the multiple-unknown equations in sequential order, which means given a list of equations $(e_1, e_2, ...e_n)$, we swap them to the order $(e_n, e_1, e_2, ...e_{n-1})$.
- **Unknown Variable Swapping *var*** Similarly, we swap the multiple unknown variables in sequential order, which means given a list of unknown variables in the equation $(x_1, x_2, ...x_n)$, we change the correspondence between them and the unknown variables in the original question, that the unknown variables in the new equation $(x_1^s, x_2^s, ...x_n^s)$ follows x_1^s denotes x_n and x_i^s denotes x_{i-1} for other indices. For example, from $n_1 * x_1 + n_2 * x_2 = 0$ to $n_1 * x_2 + n_2 * x_1 = 0$.

To incorporate the control codes for guiding the equation expression decoding, we follow studies in controlled text generation [47] and append a control code to the encoder input. We use an independent special token for each expression category as the control code, such as *add*, including *orig* for the example of the original equation expression.

Various studies have shown that natural language style control codes that serve as a description of the target text could benefit the model performance [38, 47]. We also use a description text based control code for each expression category, such as *Swap addition operands*. We use the description text *Original input* for the origin equation expression *orig* category, and also use it for inference at test stage. The detailed descriptions are shown in Table 6.1.

We use the prediction of the original equation expression control code *orig* for test inference since it has the most training examples.

6.2.2 MWP solving model

Solving multiple-unknown problems usually requires equation sets, which are challenging to generate. To tackle this problem, we follow the decoding target

Category	English	Chinese
add	Swap addition operands	加法交 换律
mul	Swap multiplication operands	乘法交 换律
sol	Solution form	以解形式表 达
equ	Swap equation order sequentially	交 换方程 组算式
var	Swap unknown variables order sequentially	交 换未知量
orig	Original Form	原始形式

Table 6.1: Description based control codes used for each category.

paradigm of Qin et al. [84], which introduces a Universal Expression Tree (UET) to represent multiple-unknown equation sets uniformly as an expression tree by using a dummy node as the head of the equation set. UET can also handle single-unknown problems in a unified manner.

For the solver model, we use two strong baseline models for experiments. For the first model, we leverage a seq2seq pre-trained language model BART [57, 95] as the solver model, which has reported promising results for text generation tasks. The encoder takes in the textual input and generates high-quality representations of the problem text. The decoder generates the UET based on these representations.

For the second model, we follow Li et al. [59] and use BERT-GTS as MWP solving model. We leverage the contextual pre-trained language model BERT as the encoder, and use a Goal-driven tree-structured MWP solver (GTS) [109] based on Long-Short-Term-Memory networks (LSTM) as the decoder.

6.3 Experiments

6.3.1 Datasets

We evaluate our proposed method on one single-unknown Chinese dataset **Math23K** [104] and two multiple-unknown datasets, **DRAW1K** [101] in English and **HMWP** [84] in Chinese. We show the statistics of overall data size, single and multiple unknown problem size, and the usage of control codes of the datasets in Figure

Model	Math23K	DRAW	HMWP
GTS [109]	75.6	39.9	44.6
G2T [115]	77.4	41.0	45.1
SAU-Solver [84]	-	39.2	44.8
BART† [95]	80.4	32.1	41.5
BERT-GTS† [59]	82.6	42.2	48.3
Controlled BART	82.3	45.3	47.9
Controlled BERT-GTS **token**	**84.0**	50.2	54.1
Controlled BERT-GTS **description**	83.3	**52.1**	**55.2**

Table 6.2: Results on MWP datasets. † denotes our implementation results. *token* denotes using special tokens as control code. *description* denotes using a piece of description text as control code.

Model	Math23K	DRAW	HMWP
BERT-GTS	82.6	42.2	48.3
+ *add*	83.0	46.8	50.8
+ *mul*	83.3	47.6	51.9
+ *sol*	-	46.3	50.5
+ *equ*	-	48.3	50.1
+ *var*	-	47.4	50.1
All	**84.0**	**50.2**	**54.1**
- *code*	83.3	49.6	49.6

Table 6.3: Ablation Study on MWP datasets. *+ control code* denotes using only one control code. *All* denotes using all control codes. *- code* denotes using the examples as data augmentation without control codes.

6.2. The five control code methods are enumerated for each example to generate new equation expressions. While *sol* is applicable for both single-unknown and multiple-unknown problems, the annotation schema in Math23K uses the Solution Form, which corresponds to *orig*, that no more further equation expressions are generated for*sol*. We use from 1.87 to 6.15 times of original data examples size for training on the three datasets.

6.4 Experimental Details

We evaluate Math23K on the standard train test setting. DRAW1K and HMWP are evaluated by 5-cross validation.

For DRAW1K, we use the bert-base pre-trained encoder. For Math23K and HMWP, we use the pre-trained encoder that could be found here [1].

For Math23K, the max text length is 256, the max equation decoding length is 45, the batch size is 16 and the epochs number is 50. We use AdamW with a learning rate of 5e-5.

For DRAW1K, the max text length is 256, the max equation decoding length is 32, the batch size is 16 and the epochs number is 50. We use AdamW with a learning rate of 5e-5.

For HMWP, the max text length is 1024, the max equation decoding length is 100, the batch size is 8 and the epochs number is 50. We use AdamW with a learning rate of 5e-5.

Experiments are conducted on NVIDIA 3090 and A100(80G). The runtime for the longest experiments is around 6 hours.

6.4.1 Results

We show our experimental results on the three datasets in Table 6.2. We compare our results with three models: **GTS** uses an LSTM encoder and decoder, which considers tree structure information during decoding; **G2T** uses a Graph Neural Network that considers quantity information as the encoder and similar tree

[1] https://huggingface.co/yechen/bert-base-chinese

decoder; **SAU-Solver** introduces a semantically-alignment to the target vocabulary of the equations to improve the GTS decoder. As we can see, our method outperforms the baseline for both models on all datasets. The accuracy of different models gains improvement from 1.8% to 1.9% for single-unknown problems and from 4.8% to 13.2% for multiple-unknown problems. Description text based control codes achieve better performance on multiple-unknown datasets, which have more expression categories. Such control codes could be beneficial as more controlled equation generation strategies are applied, which we leave as future work. The results demonstrate the effectiveness of our method, especially for multiple-unknown problems.

6.4.2 Ablation Study

We conduct further analysis on the more effective model BERT-GTS. In Table 6.3, we show the ablation study using different control codes. As shown in the Table, using each control code individually can improve the model's prediction. *mul* is particularly effective for all datasets since it has an extensive example size for each dataset. Using all control codes together further boosts the model performance by providing diverse mathematical reasoning logic as guidance.

We also show the results of removing the control codes and solely using the diverse equation expressions in a data augmentation manner in Table 6.3. Solely introducing diverse mathematical reasoning logic can also improve the model performance compared to the baseline model. However, the expression bias problem limits the performance since training loss could accumulate for diverse equation expressions. By incorporating control codes to guide the decoding process, our method can consider diverse reasoning logic and reduce the expression bias problem in the meantime.

6.4.3 Study on Variable Size

We show the performance on different given variable sizes of the BERT-GTS baseline model and our controlled equation generation method on Math23K in Figure 6.3. As the variable size grows, the problem becomes more complex, and the performance gap between our method and the baseline becomes more significant.

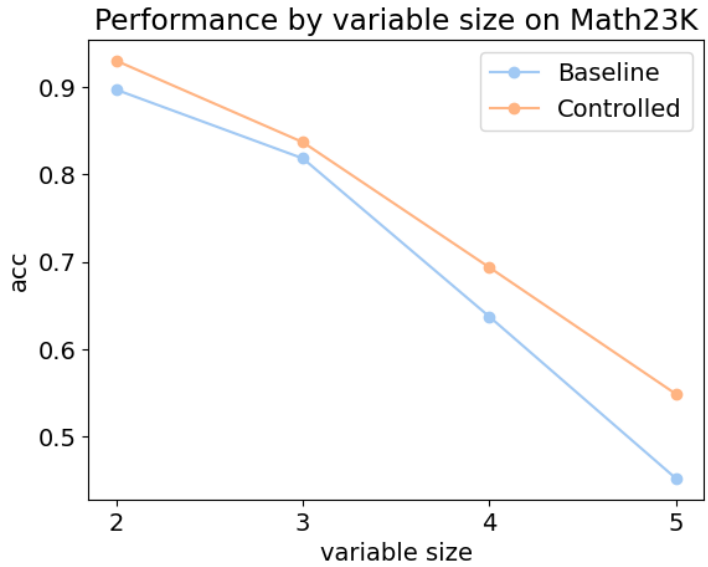


Figure 6.3: Performance on different given variable sizes.

Our method can incorporate diverse equation expressions to help the model learn mathematical reasoning logic.

6.5 Summary of This Chapter

In this chapter, we introduce diverse mathematical reasoning logic to the seq2seq MWP solver framework using five control codes to guide the solver to predict the corresponding equation expression in a controlled equation generation manner. The approach allows the solver to benefit from diverse reasoning logic beyond the human-annotated fixed solution equation. Meanwhile, the controlled equation generation training paradigm reduces the expression bias problem caused by diverse equation expressions. Experimental results show the effectiveness of our method, outperforming strong baselines on single-unknown (Math23K) and multiple-unknown (DRAW1K, HMWP) datasets.

Chapter 7

ComSearch: Equation Searching with Combinatorial Strategy for Solving Math Word Problems with Weak Supervision

7.1 Introduction

Despite the challenges posed by mathematical equivalence and the presence of numerical coincidences, researchers have made efforts to address these issues in the field of numerical reasoning. Various studies have focused on different settings for the task of numerical reasoning. Traditionally, the "full supervision" setting has been employed, which involves the annotation of equation expressions. However, this approach is expensive and time-consuming.

Hong et al. [42] (LBF) and Chatterjee et al. [14] (WARM) addressed this problem and proposed the *weak supervision* setting, where only the answer value annotation is given for supervision. Such a setting forms pseudo question-candidate equation pairs, which hold the correct answer value for training with the complex-

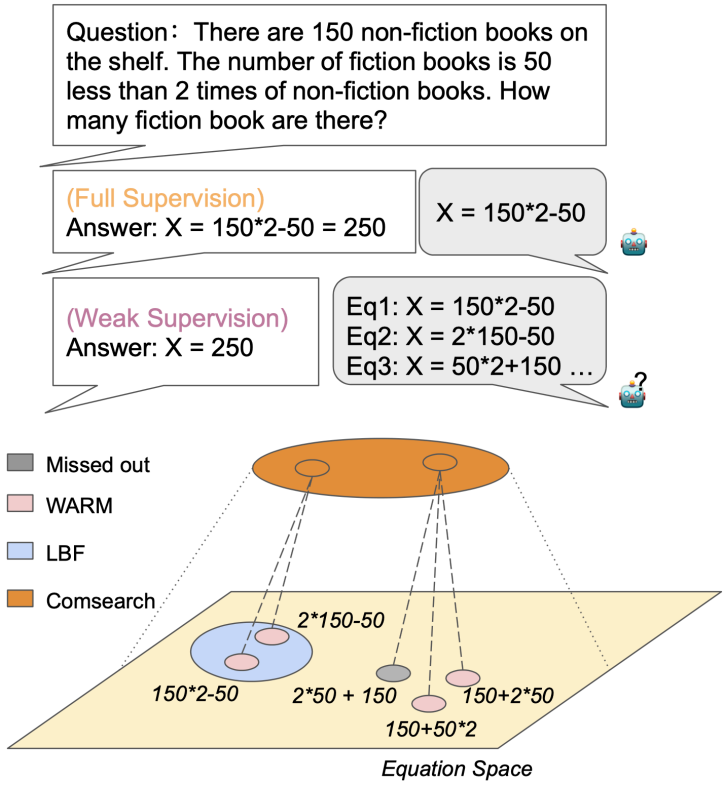


Figure 7.1: Example of MWP solving system under full supervision and weak supervision.

ity of $O(n^{2n})$ for n variables enormous possible equation space. Computational efficiently extracting such pairs becomes the major challenge since it is computationally impossible to traverse all possible equations, especially when the example has more variables (e.g., 88,473,600 for 6 variables). As we show in Figure 7.1, previous studies sample a limited set of equations via random walk [42] or beam searching [14]. However, the algorithms can only cover a limited part of the data, which we refer to as *recall*. As shown in Table 7.1, LBF [42] only covers 30% of the examples of more than 4 variables. Moreover, the random walk algorithm lacks robustness and leads to a high performance variance.

We observe that although the equation search space is ample, many equations are mathematically equivalent under the commutative law, associative law,

Model	≤ 3	≥ 4
LBF	88.1%	30.9%
ComSearch	94.4%	94.5%

Table 7.1: Searching result recall on problems of different variable sizes.

or other equivalent forms. Hence, searching for these equivalent equations is redundant, especially for difficult examples with a larger number of variables. For example, $a+b+c+d*e$ has 48 equivalent forms that hold the same mathematical meaning considering only the commutative law. Eliminating such redundancy in the searching space could reduce computational complexity. In this paper, we propose a combinatorial-strategy-based searching method **ComSearch** that enumerates *non-equivalent equations* without repeating, which can robustly extract candidate equations for a wide range of unlabeled data and build a high recall pseudo data with equation annotation even for difficult examples. To this end, the main idea of Comsearch is to use depth-first search (DFS) to enumerate only one representative equation for each set of equivalent equations and then check whether the equation holds the correct answer value. Comsearch effectively compresses the searching space, e.g., up to 111 times for 6 variables compared to bruce-force searching. As shown in Table 7.1, ComSearch can achieve a relatively high recall for different variable sizes. Our method could be proven to have lower approximate complexity.

While Comsearch only searches among *non-equivalent equations*, we observe that many examples still have multiple candidate equations through which we can get the final answer. As shown in Figure 1, Equation 1 (Eq1: X=150*2-50) and Equation 3 (Eq3: X=50*2+150) can get the same value, but Equation 3 holds a false mathematical reasoning logic, and using Equation 3 as the pseudo label would bring in noise. We address this data noise as the *false-matching* problem, which has been ignored in previous studies, since their methods do not consider whether the multiple candidate equations of one example are caused by equivalent equation forms or false matching. To address this problem, we investigate how the false-matching problem drags down the system's performance and propose two ranking models to alleviate this problem. For examples with multiple candidate

equations from ComSearch, the ranking module first collects a set of candidate equations, then assign a score by a draft model trained on pseudo data with only a single candidate equation to each candidate to choose the best pseudo label. In addition to candidates from the searching result of ComSearch, we observe that beam search results of the draft model can also serve as a high-precision candidate equation. We investigate these two settings for candidate equation sets.

We conduct experiments on two strong MWP solvers, achieving state-of-the-art (SOTA) results under the weakly supervised setting, especially for examples with many variables. The results also demonstrate the effectiveness and generalization ability of our method.

In summary, our contribution is three-fold:

- We propose ComSearch, a searching algorithm that enumerates non-equivalent equations without repeat to search candidate equations effectively.
- We are the first to investigate the *false-matching* problem that brings noise to the pseudo training data. We propose a ranking module to reduce the noise and give a detailed oracle analysis of the problem.
- We perform experiments on two MWP solvers with our ranking module and achieve SOTA performance under weak supervision.

7.2 Methodology

We show the pipeline of our method in Figure 7.2. Our method consists of three modules: the Search with combinatorial strategy (**ComSearch**) module that searches for candidate equations; the MWP model that is trained to predict equations given the natural language text and pseudo labels; the Ranking module that uses an explorer model to find candidate equations and select the best candidate equation with a scoring model.

7.2.1 ComSearch

Directly searching for non-equivalent equation expressions is difficult because the searching method needs to consider Commutative law, Associative law, and other

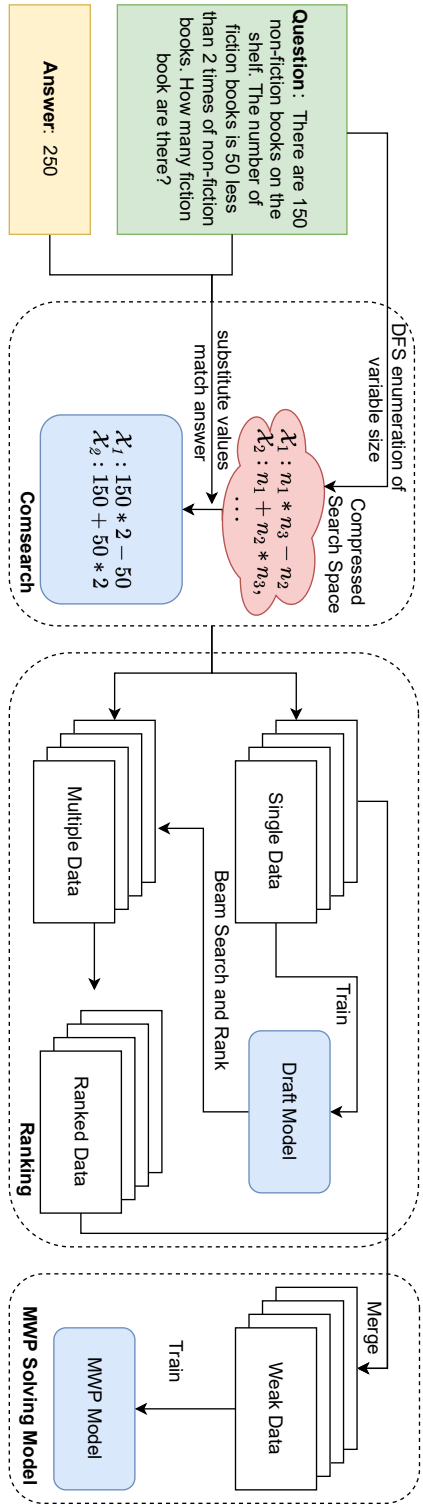


Figure 7.2: The model overview.

equivalent forms. We show how equivalent equations could be merged into a representative form $\mathcal{X}$, and the enumeration of $\mathcal{X}$ can transverse all non-equivalent equations for four arithmetic operations.

We define the set of *non-equivalent equations* using four arithmetic operations as S_n. We first split the equations to two categories, either $S^{\pm}$ where the outermost operators are $\pm$, such as $n_1/n_2+n_3-n_4$ and $n_1/n_2-(n_3-n_4)$, or S^{*} where the outermost operators are $*$, such as $(n_2+n_1)*(n_3-n_4/n_5)$. We call the former a *general addition equation* and the latter a *general multiplication equation.*

$$S_m^{\pm} = \{(n_1 * (..)) \pm (n_i * (..)) \pm ..n_m\} \tag{7.1}$$

$$S_m^{*} = \{(n_1 \pm (..)) * (n_i \pm (..)) * ..n_m\} \tag{7.2}$$

These two sets are symmetrical, so we only need to consider one set. Consider elements in $S_m^{\pm}$, we can rewrite the equation to the representative form $\mathcal{X}$:

$$\begin{aligned}\mathcal{X} =&((n_i * (..)) + (n_j * (..)) + ..) \\ &- ((n_k * (..)) + (n_l * (..)) + ..)\end{aligned}$$

For example, $n_1/n_2 - n_3 + n_4$ and $n_1/n_2 - (n_3 - n_4)$ are equivalent, that they are both rewritten as $(n_1/n_2 + n_4) - n_3$. $(n_2 + n_1) * (n_3 - n_4/n_5)$ could be rewritten as $(n_1 + n_2) * (n_3 - n_4/n_5)$. Trivially, any two equations that are represented by the same $\mathcal{X}$ are equivalent. We give proof of the number of inequivalent expressions involving n operands in Appendix Section 7.4, which shows that any two equivalent equations are written as the same $\mathcal{X}$. Thus the enumeration of $\mathcal{X}$ is equivalent to the enumeration of non-equivalent equations. The enumeration problem of these equations is an expansion of solving Schroeder's fourth problem [94], which calculates the number of labeled series-reduced rooted trees with m leaves.

Considering elements in $S_n^{\pm}$, we can rewrite the equation to x. Thus we can form a mapping $g : x \to g(x)$ from a general addition equation x to a skeleton structure expression $g(x)$. :

$$
\begin{aligned}
x =&((x_i \circledast (..)) + (x_j \circledast (..)) + ..) \\
&- ((x_k \circledast (..)) + (x_l \circledast (..)) + ..) \\
g(x) =&(x_i(..))(x_j(..))..\&(x_k(..))(x_l(..))..
\end{aligned}
$$

The order of x_i within the same layer of brackets is ignored in $g(x)$, it can deal with the equivalence caused by Commutative law and Associative law. The addition and subtraction terms are split by &, that which can deal with equivalence caused by removing brackets. $g(x)$ is a bijection, so the enumeration problem transforms to finding such skeletons:

$$
\begin{aligned}
n = 1 &: a \\
g^{-1} &: a \\
n = 2 &: ab, a\&b, b\&a \\
g^{-1} &: a + b, a - b, b - a \\
n = 3 &: abc, a\&(b\&c), (ab)\&c, ... \\
g^{-1} &: a + b + c, a - (b/c), (a * b) - c, ... \\
&... \\
\end{aligned}
$$

The enumeration problem of these structures is an expansion of solving Schroeder's fourth problem [94], which calculates the number of labeled series-reduced rooted trees with n leaves. We use a deep-first search algorithm shown in Algorithm 3 to enumerate these skeletons. It considers the position of the first bracket and then recursively finds all possible skeletons of sub-sequences of the variable sequence $\mathcal{X} = \{x_k\}_{k=1}^{i}$ [103].

While considering such skeletons could enumerate all unique expressions, equations have at least one element on the left of & in our target domain and do not start with $-$ or $\div$. We further extend the algorithm to consider these cases. To be noticed, because there is at least one $+$ or $*$ operator for each equation, the left side of & must not be empty while the right part has no restrictions. Thus we define the *unit_skel(i)* equation to return possible skeletons with only one or none

Algorithm 3 *enum_skel(n)*

Require: $n \geq 1$

```
Initialize empty list skills
for i ≤ n; i = 1; i++ do
    left_list = unit_skel(i)
    right_list = enum_skels(n − i)
    for left in left_list do
        for right in right_list do
            move the start index of right to i
            new_skels += left + right
        end for
    end for
    skels += new_skels
end for
return skels
```

& and no brackets. This constraint is equivalent to finding non-empty subsets and their complement of the variable sequence $\mathcal{X}$. We can use Algorithm 3 to perform the enumeration of such skeletons, except for defining two different *unit_skel(i)* to support the enumeration of subtraction and division operation. The enumeration algorithm of non-empty subsets is trivial and omitted here.

$$unit\_skel_{div}(i) = \{(A\&\overline{A}) | A \subseteq \mathcal{X}; A \neq \emptyset\} \tag{7.3}$$

$$\begin{aligned} &unit\_skel_{sub}(i) = \\ &\quad \{((a(A-a))\&\overline{A-a}) | A \subseteq \mathcal{X}; a \in A\} \end{aligned} \tag{7.4}$$

We transform the skeletons back to equations to obtain all non-equivalent equations S_n. Such enumeration considers absolute values and omits pairs of solutions that are opposite to each other. To search effectively, for the equations that contain subtraction, we add their opposite equation to the searching space.

Given the compressed search space, we substitute the values for variables in the equation templates and use the equations of which value matches the answer number as candidate equations. If no equations could be extracted by using all

numbers, we continue to consider: (1) omitting one number, (2) adding constant number 1 and π, and (3) using one number twice. If the algorithm extracts candidates at any stage, the further stages are not considered since it would introduce repeating equations, e.g., $1 * (a + b)$ is a duplication of $a + b$.

7.2.2 MWP Solving Models

Goal-driven Tree-structured Solver We follow Hong et al. [42] and Chatterjee et al. [14] and use Goal-Driven Tree-Structured MWP Solver (GTS) [109] as the MWP model. GTS is a seq2seq model with the attention mechanism that uses a bidirectional long short term memory network (BiLSTM) as the encoder and LSTM as the decoder. GTS also uses a recursive neural network to encode subtrees based on its children nodes' representations with the gate mechanism. With the subtree representations, this model can well use the information of the generated tokens to predict a new token.

Graph-to-Tree Solver Following Chatterjee et al. [14], we conduct experiments on Graph-to-Tree Solver (G2T) [115] . G2T is a direct extension of GTS, which consists of a graph-based encoder capturing the relationships and order information among the quantities.

7.2.3 Ranking

While ComSearch enumerates equations that are non-equivalent without repeat, some variable sets can coincidentally form multiple equations with the same correct value, as shown in Figure 7.2. The equations $150 * 2 - 50$ and $150 + 50 * 2$ are non-equivalent. However, their values are equal, while only $150 * 2 - 50$ is the correct solution. We refer to this problem as *false-matching*, an important issue that previous studies have overlooked. While previous studies also collect multiple candidate equations for one example, they cannot differ whether the issue is caused by equivalent forms of the equations or *false-matching*, and they do not perform any processing on these *false-matching* examples, which brings in noise to the pseudo data.

To process these data that have multiple candidate equations, we propose two ranking methods to choose the best candidate equation for each example. The module first collects a set of candidate equation that holds the correct annotated answer value and then score the candidates to choose the pseudo label for the sample.

Before ranking, we train a draft model S on the single-candidate pseudo data because the single-candidate data is relatively reliable with fewer false-matching examples. In the first ranking method *Basic Ranker*, for a data example x, we rank among the multiple search results of Comsearch $\{y^{eq}\}^{search}$. Then we use the draft model S to calculate the conditional probability of y^{eq} at each time step t. The score of the length k equation s^{eq} is defined as:

$$s^{eq} = \sum_{t=0}^{k} log(S(x, y_t^{eq})) \tag{7.5}$$

We use the candidate equation that has the highest score as the pseudo label of this example.

Empirically, we observe that performing beam search on the draft model S could also generate high-precision candidate equations. Thus in the second method *Beam Ranker*, we further explore more candidate equations with beam search. We add beam search predictions of S that hold the correct value $\{y_{eq}\}^{beam}$ to the candidate equation set along with Comsearch results $\{y^{eq}\}^{search}$. The score function is defined the same as the basic ranker.

7.3 Analysis on ComSearch

7.3.1 Search Statistics

We give statistics of ComSearch in Table 7.2. Among the 23,162 examples, 233 have more than 6 variables that we filter out, and 51 use the power operation that our method is not applicable. 94.5% of the examples find at least one equation that can match the answer value, significantly higher than WARM and LBF, which cover only 80.1% of the examples. 17,959 examples match with only one equation, and 3,947 examples match with two or more equations that need the

Model	**Term**	#	Prop(%)
-	All Data	23,162	-
Ours	Too Long	233	1.0
	Power Operator	51	0.2
	Single	17,959	77.5
	Multiple	3,947	17.0
	Data	21,906	**94.5**
WARM	Data (w/o beam)	-	14.5
	Data (w/ beam)	-	80.1
LBF	-	-	80.1

Table 7.2: Statistics of ComSearch Results.

#Variable	Bruce-Force	Brackets	Commutative	Ours	Ratio
1	1	1	1	1	1
2	8	8	6	6	1.3
3	192	144	108	68	2.8
4	9,216	5,184	3,816	1,170	7.9
5	737,280	311,040	224,640	27,142	27.2
6	88,473,600	27,993,600	19,841,760	793,002	111.6

Table 7.3: Empirical Results of Search Space Size.

#Var	1	2	3	4	5	≥6
LBF	91.5	86.8	88.8	31.1	25.0	38.4
Ours	67.0	93.4	96.4	98.1	94.4	73.8

Table 7.4: Result of recall on different variable sizes

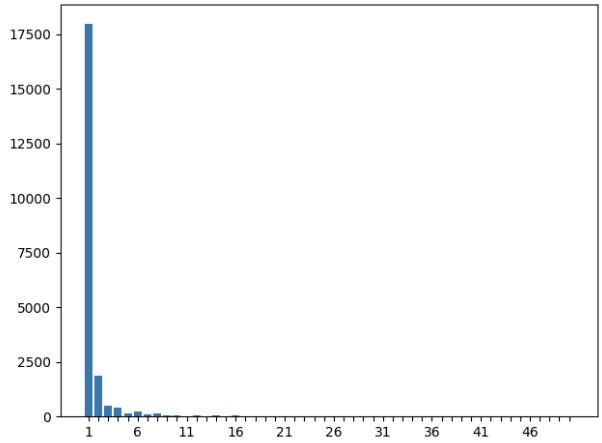


Figure 7.3: Distribution of all Candidate Equation Number.

ranking module to choose the pseudo label further. We show the distribution of these examples in Appendix Section 7.3.2.

We further break down the recall on different variable sizes in Table 7.4. As we can see, when the number of variables grows larger, the recall of LBF drastically collapses, while the recall of our method keeps steady. Sampling based methods cover only a small subset of the equation space and fail to extract candidate equations for larger variable size examples. In contrast, our method can consider a broader range of equation space, which demonstrates the superiority of our enumeration based method.

7.3.2 Distribution of Candidate Equations

The largest candidate equation number of one example is 3914. We show the distribution of candidate equations in Figure 7.3 and 7.4. The x-axis represents the number of candidates, while the y-axis represents the number of examples that

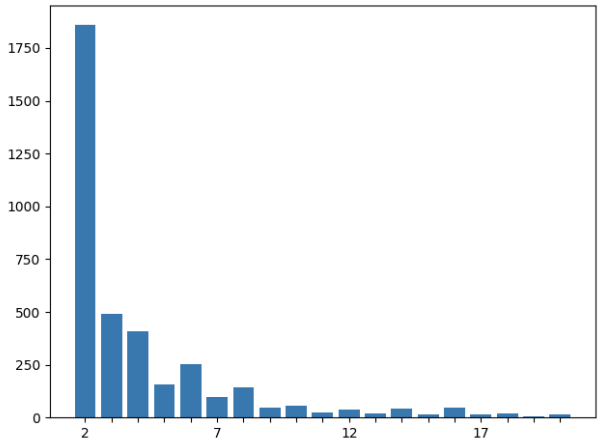


Figure 7.4: First 20 distribution of Candidate Equation Number.

have x candidate equations. We can see from Figure 7.3, which includes examples that have 1 to 50 candidates, it is a long tail distribution that most examples only have a few candidate equations. From Figure 7.4, where we zoom in and focus on examples that have 2 to 20 candidates, we can see that there are a lot of examples that have more than 2 candidate equations, and the ranking module is essential.

7.3.3 Eliminating Equivalent Equations in Search Space

We show the empirical compression of the search space with ComSearch in Table 7.3. As we can see, the compression ratio of ComSearch increases as the variable number grows, up to more than 100 times when the number of variables reaches 6. Previous studies on reducing the redundancy of equivalent expressions consider a limited set of rules, such as removing brackets [90] and Commutative Law [102]. We also show the results of considering removing brackets, where $-/\div$ can not be the children node of $+/*$, which is the compression considered in Roy and Roth [90]; and Commutative Law, which is the compression considered in Wang et al. [102]. Although the two methods can compress the search space to some extent, there is a large gap between their compression efficiency and ours, up to more than 20 times when the number of variables reaches 6.

The size of the Bruce-Force search space could be directly calculated, which is

$n! * (n-1)! * 4^{n-1}$. If we consider the exponential generating function of $card(S_n)$, based on Smooth Implicit-function Schema, we can have an approximation of S_n: $card(S_n) \sim C * n^{n-1}$, which shows our searching method compresses the search space more than exponential level. We give proof in appendix Section 7.4.

Advantages of Enumeration without repeat

The most important core of our approach is that it explicitly points out the *false-matching* problem because it can enumerate a wide range of equations while ensuring each equation holds an independent mathematical reasoning logic. Sampling methods can only sample a small set of equations that may neglect other potential candidates.

Compared to other enumeration methods, despite the enumeration efficiency, Comsearch ensures the enumeration is among non-equivalent equations, so collecting more than one candidate equation for one example shows that there exists more than one mathematical reasoning logic that could reach the annotated answer value. However, only one of the reasoning logic could be true, which elicits the *false-matching* problem. Even if we add more rules to compress the search space, as long as the non-equivalency of different equations cannot be ensured, we cannot differ *false-matching* and multiple expressions of the same mathematical reasoning logic.

7.4 Proof for Search Space Approximation

Because there is at least one $+$ or $*$ operator for each equation (i.e. $-a-b-c$ is illegal), the target S_n is not symmetric and is hard to directly approximate. We need two assisting targets to form the approximate. This proof majorly relies on Flajolet and Sedgewick [29] and [77, A140606].

We first consider target U that considers only $+$, $*$ and $\div$ three operators. We sort it into two categories: U^+ that the outermost operator is $+$ and U^* that the outermost operator is $*$. Equations such as $\frac{1}{a} * \frac{1}{b-c}$ are still considered illegal.

Z corresponds to a single variable equation. We can have the construction of U:

$$U^+ = Z + SET_{\geq}(U^*) \tag{7.6}$$
$$U^* = Z + (2^2 - 1) * SET_{=2}(U^+) \tag{7.7}$$
$$+ (2^3 - 1) * SET_{=3}(U^+)... \tag{7.8}$$

We apply symbolic method to obtain the EGF of the constructions:

$$U^+(z) = z + \sum_{k\geq 2} \frac{1}{k!}[U^*(z)]^k \tag{7.9}$$
$$= z + [e^{U^*(z)} - 1 - U^*(z)] \tag{7.10}$$
$$U^*(z) = z + \sum_{k\geq 2} \frac{2^k - 1}{k!}[U^+(z)]^k \tag{7.11}$$
$$= z + e^{2U^+(z)} - e^{U^+(z)} - U^+(z) \tag{7.12}$$

Meanwhile, we have:

$$U(z) = U^+(z) + U^*(z) - z \tag{7.13}$$

Next, we consider target T that $-a - b - c$ is considered legal. Similarly we define $T^{\pm}$ and T^*. We consider the construction:

$$T^{\pm} = 2Z + SET_{\geq}(T^*) \tag{7.14}$$
$$T^* = 2Z + 2[(2^2 - 1) * SET_{=2}(T^{\pm}/2) \tag{7.15}$$
$$+ (2^3 - 1) * SET_{=3}(T^{\pm}/2)...] \tag{7.16}$$

With symbolic method we have:

$$T^{\pm}(z) = 2z + \sum_{k\geq 2} \frac{1}{k!}[U^*(z)]^k \tag{7.17}$$
$$= 2z + [e^{T^*(z)} - 1 - T^*(z)] \tag{7.18}$$
$$T^*(z) = 2z + 2\sum_{k\geq 2} \frac{2^k - 1}{k!}[T^{\pm}(z)/2]^k \tag{7.19}$$
$$= 2z + 2e^{T^{\pm}(z)} - 2e^{T^{\pm}(z)/2} - T^{\pm}(z) \tag{7.20}$$

The illegal equations such as $-a-b-c$ in T equals the counts of $a+b+c$, which is actually U. So we have:

$$S(z) = T(z) - U(z) \tag{7.21}$$

We now have the EGF of S_n. We can sequentially compute the first few terms of this sequence:

$$1, 6, 68, 1170, 27142, 793002, 27914126, ... \tag{7.22}$$

With Smooth implicit-function schema and Stirling approximation function we have, for an EGF $y(z) = \sum_{n\geq 0} y_n z^n$, Let $G(z,w) = \sum_{m,n\geq 0} g_{m,n} z^m w^n$, thus $y(z) = G(z, y(z))$:

$$n! * [z^n]y(z) \sim \frac{c * n!}{\sqrt{2\pi n^3}} * r^{-n+1/2} \tag{7.23}$$

$$\sim \frac{c\sqrt{2\pi nr}}{\sqrt{2\pi n^3}} (\frac{1}{r})^n (\frac{n}{e})^n \tag{7.24}$$

$$= \frac{c\sqrt{r}}{n} (\frac{n}{re})^n \tag{7.25}$$

while r:

$$G(r,s) = s \tag{7.26}$$

$$\frac{\partial G(r,s)}{\partial w} = 1 \tag{7.27}$$

and c:

$$c = \sqrt{\frac{\partial G(r,s)/\partial z}{\partial^2 G(r,s)/\partial w^2}} \tag{7.28}$$

We still need the two assisting targets to perform the approximation. We have:

$$U^+(z) = e^{z+e^{2U^+(z)}-e^{U^+(z)}-U^+(z)} \tag{7.29}$$

$$- e^{2U^+(z)} + e^{U^+(z)} + U^+(z) - 1 \tag{7.30}$$

Let $G(z, w) = z + e^{2w} - e^w - ln(1 + e^{2w} - e^w)$, considering 7.26 and 7.28, r, s and c would be constant numbers.

So we have:

$$n![z^n]U^+(z) \sim \frac{c_1\sqrt{r_1}}{n}(\frac{n}{r_1 e})^n \tag{7.31}$$

Similarly we can approximate U^*, $T^\pm$ and T^*:

$$n![z^n]U^*(z) \sim \frac{c_2\sqrt{r_1}}{n}(\frac{n}{r_2 e})^n \tag{7.32}$$

$$n![z^n]T^\pm(z) \sim \frac{c_3\sqrt{r_2}}{n}(\frac{n}{r_3 e})^n \tag{7.33}$$

$$n![z^n]T^*(z) \sim \frac{c_4\sqrt{r_2}}{n}(\frac{n}{r_4 e})^n \tag{7.34}$$

So we have:

$$u_n = n![z^n]U(z) \sim \frac{(c_1 + c_2)\sqrt{r_1}}{n}(\frac{n}{r_1 e})^n \tag{7.35}$$

$$t_n = n![z^n]T(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}(\frac{n}{r_2 e})^n \tag{7.36}$$

Since $S(z) = T(z) - U(z)$, the subtraction of u_n and t_n would be our approximation. However we observe that $r_1 \gg r_3$, that u_n can be ignored. So we have:

$$s_n = n![z^n]S(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}(\frac{n}{r_2 e})^n \tag{7.37}$$

Q.E.D.

7.5 Experiments

7.5.1 Dataset and Baselines

We evaluate our proposed method on the Math23K dataset. It contains 23,161 math word problems annotated with solution expressions and answers. We only use the problems and final answers. We evaluate our method using the train-test split setting of Wang et al [102] by the three-run average.

We compare our weakly-supervised models' math word problem solving accuracy with two baseline methods.

Chatterjee et al. [14] proposed **WARM** that uses RL to train the candidate generation model with the reward of whether the value of the equation is correct. Since the reward signal is sparse due to the enormous search space, the top1 accuracy of the candidate generation model is limited, and they use beam search to search for candidates further.

Hong et al. [42] proposed **LBF**, a learning-by-fix algorithm that searches in neighbour space of the predicted wrong answer by random walk and tries to find a fix equation that holds the correct value as the candidate equation. *memory* saves the candidates of each epoch as training data.

7.5.2 Implementation Details

We run our experiments on single card GTX3090Ti, each run takes around 2-3 hours for all models. We did not perform extra hyperparameter searching and use the same hyperparameters as the public release of the two models, except for epoch number which is decided by the validation set. The code is conducted based on Pytorch.

7.5.3 Main Results and Ablation Study

We show our experimental results in Table 7.5. We reproduced the results of LBF with their official code and found that LBF+memory lacks robustness. As we can see in the table, the performance of LBF has high variance on both the validation and test set. For a fair comparison, we additionally ran 5-fold cross-validation setting according to [42] for our model and LBF+memory with the GTS model. The results show that LBF + memory achieves a cross-validation score of 56.3% with a variance of ± 6.2, while our model achieves a cross-validation score of 59.7% with a variance of ± 1.0, which performs similar to the train-test setting. We observe that its performance highly relies on the initialization of the model. When fewer candidates are extracted at early-stage training, the performance drops drastically since LBF relies on random walks in an enormous search space. Our method achieves state-of-the-art performance and outperforms

Model	**Valid(%)**	**Test(%)**
GTS based		
WARM	-	12.8
+beam	-	54.3
LBF†	57.2(±0.5)	55.4(±0.5)
+memory†	56.6(±6.9)	55.1(±6.2)
Ours†	**61.0**(±0.3)	**60.0**(±0.3)
Supervised†	-	75.6
G2T based		
WARM	-	13.5
+beam	-	56.0
Ours†	**61.7**(±1.1)	**60.5**(±0.6)
Supervised†	-	77.4

Table 7.5: Results on Math23K. ± denotes the variance of 3 runs for valid/test. *Supervised* denotes full supervision upper bound. † denotes the results of our implementation, other results are from the original paper.

other baselines up to 3.8% and 2.7% on train-test and cross-validation settings. Our method is also more robust with minor variance.

We perform an ablation study with the GTS-based train-test setting in Table 7.6. *Single Equation* denotes using the 17,959 examples that only match with one equation, the model achieves 57.5% performance, which is slightly lower than using all data and the ranking module, outperforming other baseline models. The result shows that the examples with only one matching could be considered highly reliable and achieve comparable performance with a smaller training data size. We observe a performance drop of at least 2.9% without the ranking module, showing that our ranking module improves the performance. We observe a performance gap of 0.9% between the two rankers, demonstrating the importance of considering candidate equations from the model prediction.

Model	Valid(%)	Test(%)
Proposed Method	61.0	60.0
w/o Multiple Data	58.9	57.5
w/o Ranking	57.3	56.3
w/o Beam Search	60.1	59.2

Table 7.6: Results of Ablation Study for Ranking. 'w/o Multiple Data' denotes only using single candidate pseudo data for training. 'w/o Ranking' denotes removing the ranking module and randomly sampling an equation for the examples that match with two or more equations. 'w/o Beam search' denotes using the basic ranker for ranking.

Model	Micro Eq Acc(%)
Single	81.4
Multiple	2.7
All Data	23.0
Basic Ranker(Multiple)	45.6
Beam Ranker(Multiple)	47.7
Beam Ranker(All Data)	76.3

Table 7.7: Equation accuracy of different methods. 'All Data' denotes considering both the single and multiple data.

7.5.4 Analysis

We conduct analysis on GTS train-test setting since the model achieves similar performance compared with G2T and the run time is less.

Oracle Test

While our searching method covers 94.5% of the training data, as shown in Table 7.2, there is still a significant performance gap of more than 15% between the weakly supervised performance and fully supervised performance, as shown in Table 7.5. As stated in Section 7.2.3, we observe that the *false-matching* problem could potentially draw down the performance, which is verified by the effectiveness

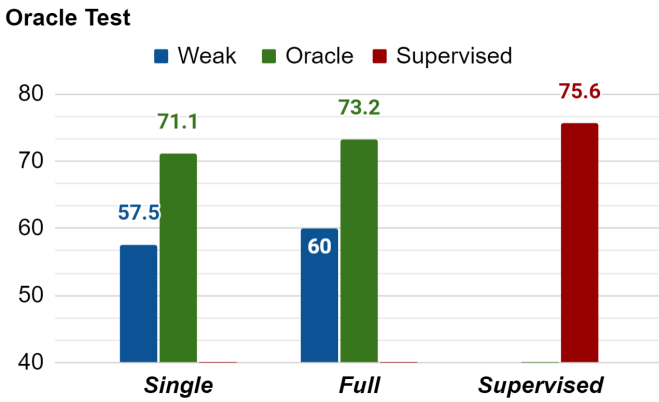


Figure 7.5: Results of Oracle Test with gold labels.

of the ranking module.

To further analyze our two modules, we perform two oracle tests for the weakly supervised system. In Figure 7.5, using the same data examples, we replace the weakly supervised annotations with the supervised gold labels and train the MWP solver. We can observe a performance gap of around 10% using the same data examples as training data, which indicates that the weakly supervised annotations contain noise. Since all candidate equation annotations have the correct answer, the *false-matching* problem is why this noise exists. The results show that the *false-matching* problem is the critical issue in the weakly supervised setting that causes the performance gap compared to supervised setting.

To investigate the noise in the pseudo training data, we perform an oracle analysis of the *Micro Equation Accuracy* of the pseudo training data. *Micro Equation Accuracy* is defined by what proportion of training instance holds the correct equation solution, which means the instance is not a *false-matching* example. In Table 7.7, we show the results of micro equation accuracy of the training data. We check whether the pseudo equation annotations that our system obtains are equivalent to the gold labels for each instance. We can see that even in the **Single** data that can only extract one candidate equation, the micro equation accuracy shows there is still noise in the pseudo training data. We show examples in the case study section to explain this problem. The examples that extract more than one candidate have an equation accuracy rate as low as 2.7%, which makes

Text	Candidates	Gold	Ans
Some children are planting trees along a road every 2 meters. They plant trees on both ends of the road. At last they planted 11 trees. How long is the road?	2*(11-1)	(11-1)*2	20
A library has 30 books. On the first day, $\frac{1}{5}$ of the books were borrowed out. On the second day, 5 books were returned. How many book are there in the library now?	30 - $\frac{1}{5}$ * 5	30*(1-($\frac{1}{5}$)) + 5	29
Peter and a few people are standing in a line, one person every 2 meters. Peter found that there are 4 people before him and 5 people after him. How long is this queue?	4*5-2 , (4+5)*2	4*2 + 5*2	18

Table 7.8: Case study of ComSearch. The blue color denotes that the candidate is true-matching and the light red color denotes that the candidate is false-matching.

our ranking system essential. Benefiting from the ranking system, the multiple candidate data can achieve a higher equation accuracy rate. The Beam ranker performs better than the basic ranker considering beam search results.

Case Study

We conduct a case study for ComSearch on three examples to further discuss the strengths and limitations of the method in Table 7.8. The first example extracts only one candidate equation; although the written expression is different from the gold label, the two equations are equivalent, and the candidate is true-matching. The second example extracts only one candidate equation; the *false-matching* candidate coincidentally equals the correct answer with this set of variable numbers. However, the candidate expression and gold label expression are not equivalent. The algorithm reaches a candidate at the stage of using all numbers and does not further search for candidates that use the constant number 1. The third example

	Train				Test		
	Micro Eq Acc(%)		Macro Eq Acc(%)		Ans Acc(%)		
#Var	LBF	Ours	LBF	Ours	LBF	Ours	Prop(%)
1	91.8	**96.3**	**88.2**	64.9	**75.0**	50.0	1.6
2	82.9	**94.8**	78.1	**88.7**	**75.2**	73.4	33.1
3	54.2	**78.9**	57.4	**76.1**	56.2	**62.9**	48.5
4	38.0	**58.0**	13.6	**57.4**	4.8	**25.8**	12.4
5	8.6	**31.1**	4.2	**29.4**	3.2	**16.1**	3.1
≥ 6	5.1	**50.6**	1.2	**38.1**	0	**30.1**	1.3

Table 7.9: Results of different variable sizes.

extracts two candidate equations, while only $(4 + 5) * 2$ holds the correct mathematical knowledge. The two candidates appear at the same searching stage, and such *false-matching* cannot be avoided by Comsearch, where we need the ranker to help filter out the *false-matching* noise. In this example, the two rankers both select the correct label.

Study on Number of Variables

The distribution of different variable size instances in Math23K dataset is imbalanced, so we further break down the performance of different variable sizes compared with LBF in Table 7.9. The *Micro Equation Accuracy* shows our method can extract higher quality pseudo data for all variable sizes compared to previous sampling based methods, especially for examples with more variables.

The recall of candidate extraction methods is another important factor that affects performance. Therefore, in addition to *Micro Equation Accuracy*, we further investigate the *Macro Equation Accuracy* of the two methods, which is defined as equation accuracy on an average of each math word problem. We show that, except for 1 variable, our method has significant advantages over LBF, especially for difficult examples. This demonstrates that our method can effectively extract high equality data of a large quantity. We also show the test answer accuracy of our method and LBF of different variable sizes, which positively correlates with the *Macro Equation Accuracy*. Eliminating equivalent equations allows our

method to consider the larger search space, while sampling based methods such as LBF limit to a small neighbour space of the model prediction. When the variable number is small, the in-place random walk of LBF can possibly reach the correct equation, that for the examples with 1 or 2 variables, LBF has a slight performance advantage. When the variable number grows larger, as shown in Table 7.3, the gap between the efficiency of our searching method and LBF expands, and our method can consider more equations candidates and achieve higher recall and better recall performance. Moreover, the *false-matching* problem is more severe when there are more variables; ignoring the problem would cause low *Micro Equation Accuracy* and bring in more noise to the pseudo training data.

7.6 Study on Large Language Models

Recent studies have investigated leveraging large language models for solving math word problems [105]. Chain-of-Thought prompting [105] uses the prompt of *Let's think step-by-step* to force the model to generate a rationale of the question before predicting the answer. Few-shot prompting [85] feeds in a set of data examples to the model before predicting the result. In Table 7.10 we show the results of using weak supervised pseudo candidate equations extracted by ComSearch as annotation for few shot prompts on Math23K with GPT-3 text-davinci-002. Although Math23K is an Chinese dataset, due to the relatively poor multilingual ability of GPT-3, using English prompts to force the model to generate English rationale can achieve better results. As we can see, using weakly supervised examples for few shot prompting can greatly boost performance compared to few-shot baseline. The performance is also comparable with using gold examples. By extracting the equation predicted by the LLM and using a calculator for the final answer, the performance could be further boosted. Under such setting, the performance of using weak examples and gold examples is still comparable.

Model	Acc(%)
Zero-shot	10.1
Zero-shot Chinese CoT	16.5
Zero-shot English CoT	26.9
Few-shot	24.0
Few-shot Weak CoT	31.5
Few-shot Gold CoT	32.0
Few-shot Weak with calculator	44.1
Few-shot Gold with calculator	45.2

Table 7.10: Results of Large Language Models performance on Math23K. CoT denotes Chain-of-Thoughts prompting.

7.7 Related Work

Early approaches to solving math word problems mainly depend on hand-craft rules and templates [7, 13]. Later studies either rely on semantic parsing [92, 97, 121], or try to obtain an equation template [51,56,90,91]. Recent studies focus on using deep learning models to predict the equation template for full supervision setting.

For weakly supervised setting, Hong et al. [42] and Chatterjee et al. [14] suffers from two major drawbacks. First, they apply equation candidate searching on an enormous searching space, while our method can effectively extract high-quality candidate equations. Hong et al. [42] results in low robustness and low performance on examples with more variables. Chatterjee et al. [14] results in low coverage of examples that can extract candidate equations. Second, they use all candidate equations for training and neglect the *false-matching* problem, which is the key issue that drags down the model performance in weakly supervised setting, while our ranking module addresses this issue and further boosts the performance.

To eliminate equivalent expressions, Roy and Roth [90] proposed a model that decomposes the equation prediction problem into various classification problems, eliminating some equivalence forms of the equation. However, the compression is highly integrated with their model and cannot generalize to other models, in-

cluding the SOTA seq2seq based models. Moreover, it can only cover limited equivalence forms, leaving out various important forms such as Commutative law and Associative law. [102] proposed a normalization method for supervised MWP systems that considers Commutative law. The method merges several equivalent expressions into one expression, resulting in the compression of the target equation space. However, their method requires bruce-force enumeration before compression, which remains to have high computational complexity. Only limited equivalent forms are considered in both studies, and the equation space is still considerably ample.

Various studies [54, 71] in ARQMath competition [70] and NTCIR benchmark [113] have investigated the math retrieval task that retrieves the most related mathematical passage for a question, which have clear semantic meanings given by the textual description. In our ranker setting, the scoring targets, i.e., plane mathematical equations, cannot provide the semantic meanings that contextual embedding similarity based methods used in math retrieval benchmarks require. With fully supervised training data, retrieval-based methods only achieve 40% accuracy [104] on Math23K.

Spurious programs in weakly supervised semantic parsing is a close analogy of the false-matching problem, which refers to incorrect programs that lead to correct denotations. The major difference is that the function names of the spurious programs are natural language defined, so the programs have semantic meanings. Extra knowledge bases [6] and lexicon clues [35] were used to denoise the spurious programs, which is not applicable for complex lexicon patterns MWPs that the solution equation uses operators '$+, -, *, /$' that have no semantic meaning. Pasupat and Liang [79] uses a small human-annotated dataset for denoising. Guu et al. [37], which proposes a re-weighted optimization loss for the examples. However, their method relies heavily on hyperparameter tuning and gains negative results on many datasets. Thus these methods are not suitable for the setting in our paper.

7.8 Summary of This Chapter

In this chapter, we propose ComSearch, a searching method based on a Combinatorial strategy, to extract candidate equations for Solving Math Word Problems under weak supervision. ComSearch compresses the enormous search space of equations beyond the exponential level, allowing the algorithm to enumerate all possible non-equivalent equations to search for candidate equations. We investigate the *false-matching* problem, which is the critical issue that drags down performance, and propose a ranking model to reduce noise. Our experiments show that our method obtains high-quality pseudo data for training and achieves state-of-the-art performance under weak supervision settings, outperforming strong baselines, especially for examples with more variables.

Chapter 8

Conclusion

8.1 Overview

In this thesis, we addressed the unique challenges posed by numerical reasoning, a multi-hop question answering task characterized by multiple mathematically equivalent solutions. We identified three key challenges and proposed innovative solutions to overcome them.

In Chapter 3, we introduced the complexity of mathematical equivalence, which makes it difficult for the general crowd to annotate high-quality samples. To address this, we proposed reverse operation based data augmentation for mathematical word problem (MWP) solving. Inspired by human double-checking during calculations, this method performed cheap and accurate data augmentation that could be adapted to any model. The augmented data not only improved performance but also provided supervision of new mathematical knowledge points. Experimental results on the Math23K dataset demonstrated the state-of-the-art performance of our method compared to a strong baseline.

Chapter 4 focused on enhancing the model's ability to learn the underlying reasoning process in numerical reasoning tasks. We proposed three solution program-centric auxiliary pretraining tasks at both the whole-program and sub-program levels. These tasks guided the model to distinguish relevant and irrelevant variables, predict operators, and mask keyphrases. Our experimental results on the FinQA and MultiHiertt datasets showcased substantial improvements across dif-

ferent scales of pre-trained language models (PLMs), highlighting the potential of leveraging program annotations for future research.

Chapter 5 presented the Textual Enhanced Contrastive Learning framework, which leveraged both supervised and self-supervised supervision to enhance the model's understanding of contextual information and bridge textual variance to mathematical logic. We introduced novel data augmentation methods and a two-stage retrieval approach to enrich the candidate pool for contrastive learning. Experimental results on benchmark datasets in both English and Chinese showcased the effectiveness of our method, which achieved state-of-the-art results on the Math23K dataset.

In Chapter 6, we introduced diverse mathematical reasoning logic to the seq2seq MWP solver framework using control codes. This approach enabled the model to benefit from diverse reasoning logic beyond fixed solution equations, reducing expression bias. Experimental results on single-unknown and multiple-unknown datasets demonstrated the effectiveness of our method, outperforming strong baselines.

Finally, in Chapter 7, we introduced ComSearch, a searching method based on a Combinatorial strategy, to extract candidate equations for solving Math Word Problems under weak supervision. ComSearch effectively compressed the enormous search space, enabling the enumeration of all possible non-equivalent equations to search for candidate equations. We also addressed the false-matching problem by proposing a ranking model to reduce noise. Experimental results demonstrated that our method obtained high-quality pseudo data for training and achieved state-of-the-art performance, outperforming strong baselines, particularly for examples with multiple variables.

Overall, this thesis contributed innovative solutions to address the challenges posed by numerical reasoning tasks. The proposed methods and frameworks achieved state-of-the-art performance on various datasets, highlighting their effectiveness and potential for improving mathematical problem-solving capabilities. The findings of this thesis pave the way for further research and development in the field of numerical reasoning and its applications in multi-hop question answering tasks.

8.2 Limitations and Future Studies

Rely on Supervised Data One limitation of these studies is that the effectiveness of the proposed methods heavily relies on annotated data, whether it is for data augmentation, pretraining tasks, searching algorithms, or control codes. Annotating large-scale datasets with high-quality samples is a time-consuming and labor-intensive process. Therefore, exploring techniques to reduce the dependency on annotated data or developing semi-supervised or weakly supervised approaches would be beneficial.

In the context of numerical reasoning ability, a promising avenue for addressing this limitation lies in the application of self-supervised methods. Self-supervised learning approaches leverage the inherent structure and patterns within data to learn useful representations without explicit human annotations. By capitalizing on the abundant unlabeled data available, these methods can potentially overcome the challenges associated with data annotation.

To specifically enhance the model's ability of numerical reasoning, we can construct self-supervision by integrating textual descriptions with numerical values and logical relationships. By combining these different modalities, the model can learn to understand the context and meaning behind numerical data, enabling it to perform more sophisticated numerical reasoning tasks. For example, by training the model to predict missing values or infer relationships between textual descriptions and corresponding numerical values, it can develop a stronger grasp of how numbers are interconnected within a given context. This integration of textual and numerical information in a self-supervised manner empowers the model to not only process and manipulate numbers but also comprehend their underlying semantic context, ultimately enhancing its overall numerical reasoning abilities.

Advancing Numerical Reasoning with LLMs Our current study primarily focuses on deep learning-based task-specific models for numerical reasoning. However, recent advancements in the field of NLP have demonstrated the potential of general LLMs as a universal solution towards Artificial Intelligence for General Computational tasks. Although the current performance of LLMs in numerical reasoning tasks may be lower compared to task-specific models, further

investigation into LLMs and their numerical reasoning abilities is warranted.

One important area for future research is the examination of LLMs in various settings, including multilingual scenarios. Understanding how LLMs handle numerical reasoning across different languages can shed light on their generalizability and adaptability. By evaluating their performance on multilingual tasks, we can uncover insights into the potential strengths and weaknesses of LLMs in numerical reasoning across diverse linguistic contexts.

Moreover, improving the numerical reasoning performance of LLMs should be a significant topic for the next generation of numerical reasoning research. This can involve exploring novel pretraining strategies, architecture modifications, or fine-tuning techniques specifically designed to enhance the numerical reasoning capabilities of LLMs. By addressing the current performance gap between LLMs and task-specific models, we can unlock the full potential of LLMs as powerful tools for numerical reasoning tasks.

Application for boarder domain In this thesis, we have successfully explored and demonstrated the effectiveness of our proposed methods in the domain of numerical reasoning, specifically focusing on datasets centered around basic arithmetic operations at the elementary level. However, there are promising avenues for extending our research to tackle more challenging and broader domains, such as university-level math problem solving. Notably, the recently proposed MATH dataset [41] presents an intriguing opportunity for further investigation. Despite the current low performance of existing models on this dataset, our methods from Chapter 4, involving pretraining, and Chapter 5, utilizing contrastive learning, showcase potential for application in these advanced domains without the reliance on equation parsing.

Moving forward, it is imperative to enhance the adaptability of methods that rely on equation parsers to address these more complex mathematical challenges to apply methods from Chapter 3 and Chapter 7 on more advanced datasets. Integrating more advanced math tools and techniques will be crucial in achieving higher performance on such datasets. Exploring the combination of state-of-the-art equation parsing techniques such as SymPy and integrating them into our

models will be a key focus of future research.

Furthermore, our proposed methods are not restricted to the realm of mathematics alone. We envision their application in other AI for science domains, including biology, chemistry, physics, and beyond. Adapting and fine-tuning our models to suit the specific data characteristics and problem-solving requirements of these scientific domains holds tremendous potential for advancing research in these fields.

To achieve these goals, further investigation is warranted to thoroughly evaluate the performance of our methods in more diverse and challenging scenarios. Robust experimentation on university-level math datasets and AI for science datasets will be conducted to assess the models' capabilities comprehensively. Additionally, gathering feedback from domain experts and educators will be essential in refining and optimizing the methods to cater to the practical needs of real-world applications.

In conclusion, this thesis lays the groundwork for future research in numerical reasoning and beyond. We are confident that our current methods demonstrate promising transferability to more difficult and broader domains and open the doors for novel applications in various AI for science disciplines. By addressing these challenges, we aim to contribute to the advancement of AI-driven problem-solving techniques in diverse academic and real-world scenarios.

Bibliography

[1] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, 2019.

[2] A. Asai and H. Hajishirzi. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650. Association for Computational Linguistics, July 2020.

[3] Y. Bakman. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*, 2007.

[4] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder. Joint entity recognition and relation extraction as a multi-head selection problem. *Expert Systems with Applications*, 114:34–45, 2018.

[5] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[7] D. G. Bobrow. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA, 1964.

[8] D. J. Briars and J. H. Larkin. An integrated model of skill in solving elementary word problems. *Cognition and instruction*, 1(3):245–296, 1984.

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[11] Y. Cao, F. Hong, H. Li, and P. Luo. A bottom-up dag structure extraction model for math word problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 39–46, 2021.

[12] K.-E. Chang, Y.-T. Sung, and S.-F. Lin. Computer-assisted learning for mathematical problem solving. *Computers & Education*, 46(2):140–151, 2006.

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

[14] O. Chatterjee, A. Waikar, V. Kumar, G. Ramakrishnan, and K. Arya. A weakly supervised model for solving math word problems, 2021.

[15] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020.

[16] X. Chen, P. Maniatis, R. Singh, C. Sutton, H. Dai, M. Lin, and D. Zhou. Spreadsheetcoder: Formula prediction from semi-structured context. In *International Conference on Machine Learning*, pages 1661–1672. PMLR, 2021.

[17] Z. Chen, W. Chen, C. Smiley, S. Shah, I. Borova, D. Langdon, R. Moussa, M. Beane, T.-H. Huang, B. Routledge, and W. Y. Wang. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[18] Z. Cheng, H. Dong, R. Jia, P. Wu, S. Han, F. Cheng, and D. Zhang. FORTAP: Using formulas for numerical-reasoning-aware table pretraining. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1166, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[19] Z. Cheng, H. Dong, Z. Wang, R. Jia, J. Guo, Y. Gao, S. Han, J.-G. Lou, and D. Zhang. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[20] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[21] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[22] D. Dellarosa. A computer simulation of children's arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2):147–154, 1986.

[23] X. Deng, Y. Su, A. Lees, Y. Wu, C. Yu, and H. Sun. ReasonBERT: Pre-trained to reason with distant supervision. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[25] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, 2019.

[26] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. 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 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[27] H. Fang and P. Xie. CERT: contrastive self-supervised learning for language understanding. *CoRR*, abs/2005.12766, 2020.

[28] Y. Feng, J. Zhang, G. He, W. X. Zhao, L. Liu, Q. Liu, C. Li, and H. Chen. A pretraining numerical reasoning model for ordinal constrained question answering on knowledge base. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1852–1861, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[29] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[30] C. R. Fletcher. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571, 1985.

[31] T. Gao, X. Yao, and D. Chen. Simcse: Simple contrastive learning of sentence embeddings. In M. Moens, X. Huang, L. Specia, and S. W. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics, 2021.

[32] M. Geva, A. Gupta, and J. Berant. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, 2020.

[33] M. Glass, A. Gliozzo, R. Chakravarti, A. Ferritto, L. Pan, G. P. S. Bhargav, D. Garg, and A. Sil. Span selection pre-training for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2782, Online, July 2020. Association for Computational Linguistics.

[34] T. Gokhale, P. Banerjee, C. Baral, and Y. Yang. Vqa-lol: Visual question answering under the lens of logic. *arXiv preprint arXiv:2002.08325*, 2020.

[35] O. Goldman, V. Latcinnik, E. Nave, A. Globerson, and J. Berant. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the*

56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1809–1819, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[36] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - A new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[37] K. Guu, P. Pasupat, E. Liu, and P. Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[38] J. He, W. Kryściński, B. McCann, N. Rajani, and C. Xiong. {CTRL}sum: Towards generic controllable text summarization. *arXiv*, 2020.

[39] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 9726–9735. Computer Vision Foundation / IEEE, 2020.

[40] M. Heilman and N. A. Smith. Question generation via overgenerating transformations and ranking. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA LANGUAGE TECHNOLOGIES INST, 2009.

[41] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the math dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[42] Y. Hong, Q. Li, D. Ciao, S. Huang, and S.-C. Zhu. Learning by fixing: Solving math word problems with weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4959–4967, May 2021.

[43] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533, 2014.

[44] 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, 2016.

[45] Z. Jiang, Y. Mao, P. He, G. Neubig, and W. Chen. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States, July 2022. Association for Computational Linguistics.

[46] D. Kang, T. Khot, A. Sabharwal, and E. Hovy. Adventure: Adversarial training for textual entailment with knowledge-guided examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2418–2428, 2018.

[47] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[48] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[49] E. Koci, M. Thiele, J. Rehak, O. Romero, and W. Lehner. Deco: A dataset of annotated spreadsheets for layout and table recognition. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1280–1285. IEEE, 2019.

[50] K. R. Koedinger and E. L. Suecker. Pat goes to college: Evaluating a cognitive tutor for developmental mathematics. 1996.

[51] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015.

[52] 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, San Diego, California, June 2016. Association for Computational Linguistics.

[53] 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.

[54] G. Y. Kristianto, G. Topic, and A. Aizawa. Mcat math retrieval system for ntcir-12 mathir task. In *NTCIR*, 2016.

[55] V. Kumar, R. Maheshwary, and V. Pudi. Adversarial examples for evaluating math word problem solvers. In M. Moens, X. Huang, L. Specia, and S. W. Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2705–2712. Association for Computational Linguistics, 2021.

[56] 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, 2014.

[57] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[58] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167, 2019.

[59] Z. Li, W. Zhang, C. Yan, Q. Zhou, C. Li, H. Liu, and Y. Cao. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2486–2496, 2022.

[60] C.-C. Liang, K.-Y. Hsu, C.-T. Huang, C.-M. Li, S.-Y. Miao, and K.-Y. Su. A tag-based statistical english math word problem solver with understanding, reasoning and explanation. In *IJCAI*, pages 4254–4255, 2016.

[61] Z. Liang, J. Zhang, J. Shao, and X. Zhang. Mwp-bert: A strong baseline for math word problems, 2021.

[62] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe. Let's verify step by step, 2023.

[63] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, 2017.

[64] Q. Liu, W. Guan, S. Li, F. Cheng, D. Kawahara, and S. Kurohashi. Roda: reverse operation based data augmentation for solving math word problems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1–11, 2021.

[65] Q. Liu, W. Guan, S. Li, and D. Kawahara. 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, 2019.

[66] X. Liu, P. He, W. Chen, and J. Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy, July 2019. Association for Computational Linguistics.

[67] X. Liu, Y. Wang, J. Ji, H. Cheng, X. Zhu, E. Awa, P. He, W. Chen, H. Poon, G. Cao, and J. Gao. The Microsoft toolkit of multi-task deep neural networks for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 118–126, Online, July 2020. Association for Computational Linguistics.

[68] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[69] C.-K. Looi and B. T. Tan. A cognitive-apprenticeship-based environment for learning word problem solving. *Journal of Computers in Mathematics and Science Teaching*, 17(4):339–54, 1998.

[70] B. Mansouri, A. Agarwal, D. Oard, and R. Zanibbi. Finding old answers to new math questions: the arqmath lab at clef 2020. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 564–571. Springer, 2020.

[71] B. Mansouri, D. W. Oard, and R. Zanibbi. Dprl systems in the clef 2022 arqmath lab: Introducing mathamr for math-aware search. *Proceedings of the Working Notes of CLEF 2022*, pages 5–8, 2021.

[72] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, Jan. 2017.

[73] S. Miao, C. Liang, and K. Su. A diverse corpus for evaluating and developing english math word problem solvers. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 975–984. Association for Computational Linguistics, 2020.

[74] R. Mihalcea and P. Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[75] S. Mishra, A. Mitra, N. Varshney, B. Sachdeva, P. Clark, C. Baral, and A. Kalyan. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[76] A. Mitra and C. Baral. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2144–2153, 2016.

[77] OEIS Foundation Inc. The On-Line Encyclopedia of Integer Sequences, 2023. Published electronically at `http://oeis.org`.

[78] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[79] P. Pasupat and P. Liang. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23–32, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[80] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[81] A. Patel, S. Bhattamishra, and N. Goyal. Are NLP models really able to solve simple math word problems? In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tür, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2080–2094. Association for Computational Linguistics, 2021.

[82] X. Pi, Q. Liu, B. Chen, M. Ziyadi, Z. Lin, Q. Fu, Y. Gao, J.-G. Lou, and W. Chen. Reasoning like program executors, 2022.

[83] J. Qin, X. Liang, Y. Hong, J. Tang, and L. Lin. Neural-symbolic solver for math word problems with auxiliary tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5870–5881, Online, Aug. 2021. Association for Computational Linguistics.

[84] J. Qin, L. Lin, X. Liang, R. Zhang, and L. Lin. 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, Nov. 2020. Association for Computational Linguistics.

[85] O. Ram, Y. Kirstain, J. Berant, A. Globerson, and O. Levy. Few-shot question answering by pretraining span selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3066–3079, 2021.

[86] Q. Ran, Y. Lin, P. Li, J. Zhou, and Z. Liu. NumNet: Machine reading comprehension with numerical reasoning. 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 2474–2484, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[87] A. Ravichander, A. Naik, C. Rose, and E. Hovy. Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 349–361, 2019.

[88] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In K. Inui, J. Jiang, V. Ng, and X. Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics, 2019.

[89] B. Robaidek, R. Koncel-Kedziorski, and H. Hajishirzi. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*, 2018.

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

[91] S. Roy and D. Roth. Unit dependency graph and its application to arithmetic word problem solving. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[92] S. Roy and D. Roth. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6:159–172, 2018.

[93] S. Roy, T. Vieira, and D. Roth. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13, 2015.

[94] E. Schröder. Vier combinatorische probleme. *Zeitschrift für Mathematik und Physik*, 15, 1870.

[95] J. Shen, Y. Yin, L. Li, L. Shang, X. Jiang, M. Zhang, and Q. Liu. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279, 2021.

[96] Y. Shen, Q. Liu, Z. Mao, Z. Wan, F. Cheng, and S. Kurohashi. Seeking diverse reasoning logic: Controlled equation expression generation for solving math word problems. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing*, pages 254–260, 2022.

[97] S. Shi, Y. Wang, C.-Y. Lin, X. Liu, and Y. Rui. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142, 2015.

[98] J. R. Slagle. Experiments with a deductive question-answering program. *Communications of the ACM*, 8(12):792–798, 1965.

[99] M. M. Steele and J. W. Steele. Discover: An intelligent tutoring system for teaching students with learning difficulties to solve word problems. *Journal of Computers in Mathematics and Science Teaching*, 18(4):351–359, 1999.

[100] S. S. Sundaram and D. Khemani. Natural language processing for solving simple word problems. In *Proceedings of the 12th international conference on natural language processing*, pages 394–402, 2015.

[101] S. Upadhyay and M.-W. Chang. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 494–504, 2017.

[102] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu. 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, 2018.

[103] Y. Wang. *The Math You Never Thought Of.* Posts & Telecom Press Co., Ltd., Beijing, 2021.

[104] 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.

[105] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[106] 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 6381–6387, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[107] J. L. Wheeler and J. W. Regian. The use of a cognitive tutoring system in the improvement of the abstract reasoning component of word problem solving. *Computers in Human Behavior*, 15(2):243–254, 1999.

[108] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics.

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

[110] J. Xu, M. Zhou, X. He, S. Han, and D. Zhang. Towards robust numerical question answering: Diagnosing numerical capabilities of nlp systems, 2022.

[111] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics.

[112] A. W. Yu, D. Dohan, M. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[113] R. Zanibbi, A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila. Ntcir-12 mathir task overview. In *NTCIR*, 2016.

[114] J. Zhang and Y. Moshfeghi. Elastic: Numerical reasoning with adaptive symbolic compiler. In *Advances in Neural Information Processing Systems*, 2022.

[115] 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, 2020.

[116] Y. Zhao, Y. Li, C. Li, and R. Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[117] Y. Zhao, K. Meng, G. Liu, J. Du, and H. Zhu. A multi-task dual-tree network for aspect sentiment triplet extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 7065–7074, Gyeongju, Republic of Korea, Oct. 2022. International Committee on Computational Linguistics.

[118] C. Zheng, Z. Liu, E. Xie, Z. Li, and Y. Li. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*, 2023.

[119] Z. Zhong and D. Chen. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online, June 2021. Association for Computational Linguistics.

[120] F. Zhu, W. Lei, Y. Huang, C. Wang, S. Zhang, J. Lv, F. Feng, and T.-S. Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 3277–3287, Online, Aug. 2021. Association for Computational Linguistics.

[121] Y. Zou and W. Lu. Text2math: End-to-end parsing text into math expressions. 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 5330–5340, 2019.

List of Publications

[1] Qianying Liu*[1] , Wenyv Guan*, Sujian Li and Daisuke Kawahara. 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, 2019

[2] Wenyv Guan, Qianying Liu, Guangzhi Han, Bin Wang and Sujian Li. An Improved Coarse-to-Fine Method for Solving Generation Tasks. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association(ALTA)*, pages 178–185, 2019

[3] Ranran Haoran Zhang, Qianying Liu, Aysa Xuemo Fan, Heng Ji, Daojian Zeng, Fei Cheng, Daisuke Kawahara and Sadao Kurohashi. Minimize Exposure Bias of Seq2Seq Models in Joint Entity and Relation Extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 236–246, 2020

[4] Qianying Liu, Sicong Jiang, Yizhong Wang and Sujian Li. LiveQA: A Question Answering Dataset Over Sports Live. In *Chinese Computational Linguistics: 19th China National Conference (CCL)*, pages 316–328, 2020

[5] Qianying Liu, Wenyv Guan, Tianyi Li and Sujian Li. Refining Data for Text Generation. In *Chinese Computational Linguistics: 19th China National Conference (CCL)*, pages 881–891, 2020

[6] Qianying Liu, Wenyv Guan, Sujian Li, Fei Cheng, Daisuke Kawahara and Sadao Kurohashi. RODA: Reverse Operation Based Data Augmentation for Solving Math Word Problems. In *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, pages 1–11, 2021

[7] Yibin Shen*, Qianying Liu*, Zhuoyuan Mao, Fei Cheng and Sadao Kurohashi. Textual Enhanced Contrastive Learning for Solving Math Word Prob-

[1] * denotes equal contribution.

lems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4297–4307, 2022

[8] Yibin Shen*, Qianying Liu*, Zhuoyuan Mao, Zhen Wan, Fei Cheng and Sadao Kurohashi. Seeking Diverse Reasoning Logic: Controlled Equation Expression Generation for Solving Math Word Problems. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (AACL-IJCNLP))*, pages 254–260, 2022

[9] Zhen Wan*, Qianying Liu*, Zhuoyuan Mao, Fei Cheng and Sadao Kurohashi. Rescue Implicit and Long-tail Cases: Nearest Neighbor Relation Extraction. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1731–1738, 2022

[10] Qianying Liu, Wenyv Guan, Jianhao Shen, Fei Cheng and Sadao Kurohashi. ComSearch: Equation Searching with Combinatorial Strategy for Solving Math Word Problems with Weak Supervision. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 2541–2554, 2023

[11] Qianying Liu*, Zhuo Gong*, Zhengdong Yang*, Yuhang Yang, Sheng Li, Chenchen Ding, Nobuaki Minematsu, Hao Huang, Fei Cheng, Chenhui Chu and Sadao Kurohashi. Hierarchical softmax for end-to-end low-resource multilingual speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023

[12] Qianying Liu, Dongsheng Yang, Wenjie Zhong, Fei Cheng and Sadao Kurohashi. Comprehensive Solution Program Centric Pretraining for Table-and-Text Hybrid Numerical Reasoning. In *arxiv*, 2023

Paper title:	RODA: Reverse Operation Based Data Augmentation for Solving Math Word Problems
DOI:	https://doi.org/10.1109/TASLP.2021.3126932
Issue Date	2021.11
Condition to publish:	