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

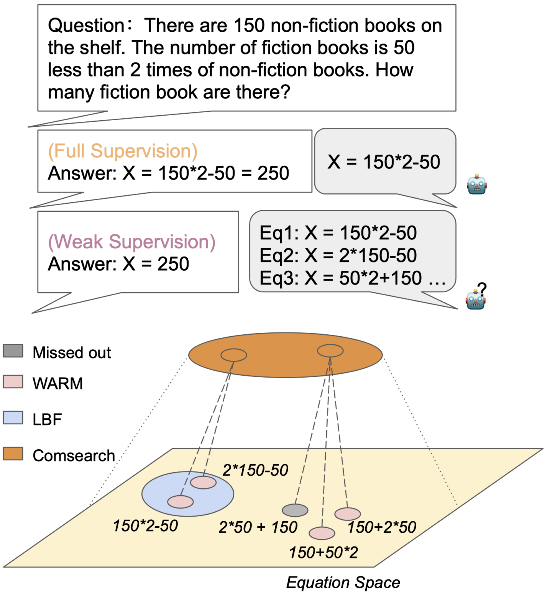
Qianying Liu[1], **Wenyu Guan**[2], **Jianhao Shen**[3], **Fei Cheng**[1] **and Sadao Kurohashi**[1]
[1] Graduate School of Informatics, Kyoto University
[2] Xiaomi AI Lab
[3] Peking University
ying@nlp.ist.i.kyoto-u.ac.jp; gwy1995@live.com; jhshen@pku.edu.cn;
{feicheng,kuro}@nlp.ist.i.kyoto-u.ac.jp;

Abstract

Previous studies have introduced a weakly-supervised paradigm for solving math word problems requiring only the answer value annotation. While these methods search for correct value equation candidates as pseudo labels, they search among a narrow sub-space of the enormous equation space. To address this problem, we propose a novel search algorithm with combinatorial strategy **ComSearch**, 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. Our approach holds a flexible framework to utilize two existing supervised math word problem solvers to train pseudo labels, and both achieve state-of-the-art performance in the weak supervision task. [1]

1 Introduction

Solving math word problems (MWPs) is the task of extracting a mathematical solution from problems written in natural language. Based on a sequence-to-sequence (seq2seq) framework that takes in the text descriptions of the MWPs and predicts the answer equation (Wang et al., 2017), task-specialized encoder and decoder architectures (Wang et al., 2018b, 2019; Xie and Sun, 2019; Liu et al., 2019; Guan et al., 2019; Zhang et al., 2020b,a; Shen and Jin, 2020), data augmentation and normalization (Wang et al., 2018a; Liu et al., 2020; Shen et al., 2022b), and pretrained models (Tan et al., 2021; Liang et al., 2021; Shen et al., 2021, 2022a) have been conducted on *full supervision* setting of the task. These settings require equation expression annotation, which is expensive and time-consuming.


Question：There are 150 non-fiction books on the shelf. The number of fiction books is 50 less than 2 times of non-fiction books. How many fiction book are there?

(Full Supervision)
Answer: X = 150*2-50 = 250

X = 150*2-50

(Weak Supervision)
Answer: X = 250

Eq1: X = 150*2-50
Eq2: X = 2*150-50
Eq3: X = 50*2+150 …

Figure 1: Example of MWP solving system under full supervision and weak supervision.

Recently Hong et al. (2021) (LBF) and Chatterjee et al. (2021) (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 complexity 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 1, previous studies sample a limited set of equations via random walk (Hong et al., 2021) or beam searching (Chatterjee et al., 2021). However, the algorithms can only cover a limited part of the data, which we refer to as *recall*. As shown in Table 1, LBF (Hong et al., 2021) only covers 30% of the examples of more than 4

[1] Our code and data is available at https://github.com/yiyunya/ComSearch

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2549–2562
May 2-6, 2023 ©2023 Association for Computational Linguistics

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

Table 1: Searching result recall on problems of different variable sizes.

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, 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 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.

2 Methodology

We show the pipeline of our method in Figure 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.

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 equivalent forms. We show how equivalent equations could be merged into a representative form $\mathcal{X}$, and the enumeration of $\mathcal{X}$ can

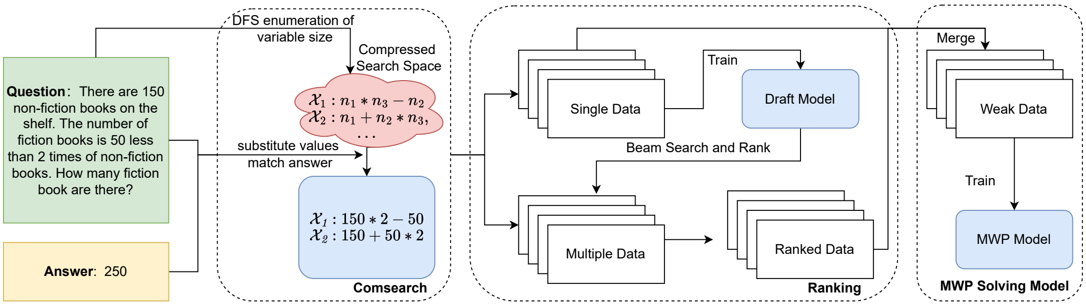

DFS enumeration of variable size
Compressed Search Space
$\mathcal{X}_1 : n_1 * n_3 - n_2$
$\mathcal{X}_2 : n_1 + n_2 * n_3,$
...
Question: There are 150 non-fiction books on the shelf. The number of fiction books is 50 less than 2 times of non-fiction books. How many fiction book are there?
substitute values match answer
$\mathcal{X}_1 : 150 * 2 - 50$
$\mathcal{X}_2 : 150 + 50 * 2$
Answer: 250
Comsearch
Single Data
Train
Draft Model
Beam Search and Rank
Multiple Data
Ranked Data
Ranking
Merge
Weak Data
Train
MWP Model
MWP Solving Model

Figure 2: The model overview.

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 $S^{\divideontimes}$ where the outermost operators are $\divideontimes$, 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*.

$$S^{\pm}_m = \{(n_1 \divideontimes (..)) \pm (n_i \divideontimes (..)) \pm ..n_m\} \quad (1)$$

$$S^{\divideontimes}_m = \{(n_1 \pm (..)) \divideontimes (n_i \pm (..)) \divideontimes ..n_m\} \quad (2)$$

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

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

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 A, 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 (Schröder, 1870), which calculates the number of labeled series-reduced rooted trees with m leaves. We give the details of the DFS in the Appendix Section D.

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 $a + b$.

2.2 MWP Solving Models

Goal-driven Tree-structured Solver We follow Hong et al. (2021) and Chatterjee et al. (2021) and use Goal-Driven Tree-Structured MWP Solver (GTS) (Xie and Sun, 2019) 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. (2021), we conduct experiments on Graph-to-Tree Solver (G2T) (Zhang et al., 2020b) . G2T is a direct extension of GTS, which consists of a graph-based encoder capturing the relationships and order information among the quantities.

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 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

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 2: Statistics of ComSearch Results.

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:

$$s^{eq} = \sum_{t=0}^{k} log(S(x, y^{eq}_t)) \tag{3}$$

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.

3 Analysis on ComSearch

3.1 Search Statistics

We give statistics of ComSearch in Table 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 ranking module to choose the pseudo label further. We show the distribution of these examples in Appendix Section B.

We further break down the recall on different variable sizes in Table 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.

3.2 Eliminating Equivalent Equations in Search Space

We show the empirical compression of the search space with ComSearch in Table 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 (Roy and Roth, 2015) and Commutative Law (Wang et al., 2018a). 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 (2015); and Commutative Law, which is the compression considered in Wang et al. (2018a). 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

#Variable	Bruce-Force	Removing Brackets	Commutative	ComSearch	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 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 4: Result of recall on different variable sizes

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 A.

3.2.1 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.

4 Experiments

4.1 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. (2018a) by the three-run average.

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

Table 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.

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

Chatterjee et al. (2021) 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. (2021) 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.

4.2 Main Results and Ablation Study

We show our experimental results in Table 5. We reproduced the results of LBF with their official code

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 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: Equation accuracy of different methods. ‘All Data’ denotes considering both the single and multiple data.

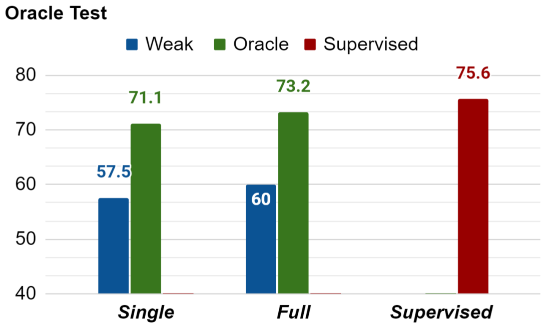


Figure 3: Results of Oracle Test with gold labels.

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 (Hong et al., 2021) 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 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 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.

4.3 Analysis

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

4.3.1 Oracle Test

While our searching method covers 94.5% of the training data, as shown in Table 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 5. As stated in Section 2.3, we observe that the *false-matching* problem could potentially draw down the performance, which is verified by the effectiveness of the ranking module.

To further analyze our two modules, we perform two oracle tests for the weakly supervised system. In Figure 3, 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*

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 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.

	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 9: Results of different variable sizes.

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, 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 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.

4.3.2 Case Study

We conduct a case study for ComSearch on three examples to further discuss the strengths and limitations of the method in Table 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 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.

4.3.3 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 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 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.

5 Related Work

Early approaches to solving math word problems mainly depend on hand-craft rules and templates (Bobrow, 1964; Charniak, 1969). Later studies either rely on semantic parsing (Roy and Roth, 2018; Shi et al., 2015; Zou and Lu, 2019), or try to obtain an equation template (Kushman et al., 2014; Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Roy and Roth, 2017). Recent studies focus on using deep learning models to predict the equation template for full supervision setting.

For weakly supervised setting, Hong et al. (2021) and Chatterjee et al. (2021) 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. (2021) results in low robustness and low performance on examples with more variables. Chatterjee et al. (2021) 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 (2015) 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, including 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. (Wang et al., 2018a) 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 (Kristianto et al., 2016; Mansouri et al., 2021) in ARQMath competition (Mansouri et al., 2020) and NTCIR benchmark (Zanibbi et al., 2016) 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 (Wang et al., 2017) 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 (Berant et al., 2013) and lexicon clues (Goldman et al., 2018) 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 (2016) uses a small human-annotated dataset for denoising. Guu et al. (2017), 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.

6 Conclusion

This paper proposes 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.

Limitations

As we observe from experiments, the performance gap between the most reliable weak data and oracle data is still 10%, and the noise rate in the pseudo data is still relatively high. This is caused by the stopping strategy of our searching algorithm. Because introducing constant numbers such as 1 and using variables for more than one time would cause meaningless multiple candidate equations (e.g., $n_1/n_1 * n_1$, $1 * n_1$), we search the equations at various stages: deleting one variable, adding a constant and using one variable multiple times. We stop searching when the stage ends and one equation is obtained. If a more advanced searching strategy that can consider such redundancy could be introduced, the reliability of the weak data could be further boosted.

Meanwhile, our ranking module only denoises multiple candidate equations examples, while the single data also has a volume of noise. We denoise with a simple strategy for one round because we focus on investigating the negative effects of the false-matching problem. For future work, we would consider applying more advanced learning from noise algorithms and denoise more training data.

In Table 4, the results shows a notable discrepancy in the performance of the #var = 1 when compared to other variable sizes and the baseline. This discrepancy can primarily be attributed to numerous geometrically related questions in the #var = 1 example set, such as the computation of the volume of a cube l^3 given the side length l, which is not encompassed by our current search methodology. A straightforward remedy would be to include this equation template in our search when handling #var = 1; however, we deliberately excluded it from our experiments to maintain consistency across the different variable sizes.

Acknowledgements

This work is partially supported by JST SPRING Grant No.JPMJSP2110 and KAKENHI No. 21H00308, Japan.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. *Publications Manual*. American Psychological Association, Washington, DC.

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. 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. Association for Computational Linguistics.

Daniel G. Bobrow. 1964. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA.

L. Carlitz and J. Riordan. 1956. The number of labeled two-terminal series-parallel networks. *Duke Mathematical Journal*, 23(3):435 – 445.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133.

Eugene Charniak. 1969. 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. Morgan Kaufmann Publishers Inc.

Oishik Chatterjee, Aashish Waikar, Vishwajeet Kumar, Ganesh Ramakrishnan, and Kavi Arya. 2021. A weakly supervised model for solving math word problems.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

Philippe Flajolet and Robert Sedgewick. 2009. *Analytic Combinatorics*. Cambridge University Press.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. 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. Association for Computational Linguistics.

Wenyv Guan, Qianying Liu, Guangzhi Han, Bin Wang, and Sujian Li. 2019. An improved coarse-to-fine method for solving generation tasks. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 178–185, Sydney, Australia. Australasian Language Technology Association.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. 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. Association for Computational Linguistics.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4959–4967.

W. Knödel. 1951. Über zerfällungen. *Monatshefte für Mathematik*, 55:20–27.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. Mcat math retrieval system for ntcir-12 mathir task. In *NTCIR*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. 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.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2021. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. 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.

Zhenwen Liang, Jipeng Zhang, Jie Shao, and Xiangliang Zhang. 2021. Mwp-bert: A strong baseline for math word problems.

Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Reverse operation based data augmentation for solving math word problems. *arXiv preprint arXiv:2010.01556*.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2370–2379, Hong Kong, China. Association for Computational Linguistics.

Behrooz Mansouri, Anurag Agarwal, Douglas Oard, and Richard Zanibbi. 2020. 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.

Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. 2021. 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.

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

Panupong Pasupat and Percy Liang. 2016. 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. Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

John Riordan and Claude E Shannon. 1942. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21(1-4):83–93.

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

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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

Ernst Schröder. 1870. Vier combinatorische probleme. *Zeitschrift für Mathematik und Physik*, 15.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yibin Shen, Qianying Liu, Zhuoyuan Mao, Fei Cheng, and Sadao Kurohashi. 2022a. Textual enhanced contrastive learning for solving math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4297–4307, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yibin Shen, Qianying Liu, Zhuoyuan Mao, Zhen Wan, Fei Cheng, and Sadao Kurohashi. 2022b. 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 (Volume 2: Short Papers)*, pages 254–260, Online only. Association for Computational Linguistics.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. 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.

Minghuan Tan, Lei Wang, Lingxiao Jiang, and Jing Jiang. 2021. Investigating math word problems using pretrained multilingual language models.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. 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.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Yun Wang. 2021. *The Math You Never Thought Of.* Posts & Telecom Press Co., Ltd., Beijing.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5299–5305.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. Ntcir-12 mathir task overview. In *NTCIR*.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2287–2305.

Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. 2020a. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4011–4017. International Joint Conferences on Artificial Intelligence Organization. Main track.

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

Yanyan Zou and Wei Lu. 2019. 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.

A 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 (2009) and OEIS Foundation Inc. (2023, 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 $\divideontimes$. 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:

$$U^{+} = Z + SET_{\geq}(U^{*}) \tag{4}$$

$$U^{*} = Z + (2^2-1) * SET_{=2}(U^{+}) \tag{5}$$

$$+ (2^3-1) * SET_{=3}(U^{+})... \tag{6}$$

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

$$U^{+}(z) = z + \sum_{k\geq 2} \frac{1}{k!}[U^{*}(z)]^k \tag{7}$$

$$= z + [e^{U^{*}(z)} - 1 - U^{*}(z)] \tag{8}$$

$$U^{*}(z) = z + \sum_{k\geq 2} \frac{2^k-1}{k!}[U^{+}(z)]^k \tag{9}$$

$$= z + e^{2U^{+}(z)} - e^{U^{+}(z)} - U^{+}(z) \tag{10}$$

Meanwhile, we have:

$$U(z) = U^{+}(z) + U^{*}(z) - z \tag{11}$$

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

$$T^{\pm} = 2Z + SET_{\geq}(T^{*}) \tag{12}$$

$$T^{*} = 2Z + 2[(2^2-1) * SET_{=2}(T^{\pm}/2) \tag{13}$$

$$+ (2^3-1) * SET_{=3}(T^{\pm}/2)...] \tag{14}$$

With symbolic method we have:

$$T^{\pm}(z) = 2z + \sum_{k\geq 2} \frac{1}{k!}[U^{*}(z)]^k \tag{15}$$

$$= 2z + [e^{T^{*}(z)} - 1 - T^{*}(z)] \tag{16}$$

$$T^{*}(z) = 2z + 2\sum_{k\geq 2} \frac{2^k-1}{k!}[T^{\pm}(z)/2]^k \tag{17}$$

$$= 2z + 2e^{T^{\pm}(z)} - 2e^{T^{\pm}(z)/2} - T^{\pm}(z) \tag{18}$$

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

$$S(z) = T(z) - U(z) \tag{19}$$

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

$$1, 6, 68, 1170, 27142, 793002, 27914126, ... \tag{20}$$

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 $y(z) = G(z, y(z))$:

$$n! * [z^n]y(z) \sim \frac{c * n!}{\sqrt{2\pi n^3}} * r^{-n+1/2} \tag{21}$$

$$\sim \frac{c\sqrt{2\pi n r}}{\sqrt{2\pi n^3}} (\frac{1}{r})^n (\frac{n}{e})^n \tag{22}$$

$$= \frac{c\sqrt{r}}{n} (\frac{n}{re})^n \tag{23}$$

while r:

$$G(r,s) = s \tag{24}$$

$$\frac{\partial G(r,s)}{\partial w} = 1 \tag{25}$$

and c:

$$c = \sqrt{\frac{\partial G(r,s)/\partial z}{\partial^2 G(r,s)/\partial w^2}} \tag{26}$$

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

$$U^{+}(z) = e^{z + e^{2U^{+}(z)} - e^{U^{+}(z)} - U^{+}(z)} \tag{27}$$

$$- e^{2U^{+}(z)} + e^{U^{+}(z)} + U^{+}(z) - 1 \tag{28}$$

Let $G(z,w) = z + e^{2w} - e^{w} - ln(1 + e^{2w} - e^{w})$, considering 24 and 26, r, s and c would be constant numbers.

So we have:

$$n![z^n]U^{+}(z) \sim \frac{c_1\sqrt{r_1}}{n} (\frac{n}{r_1 e})^n \tag{29}$$

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

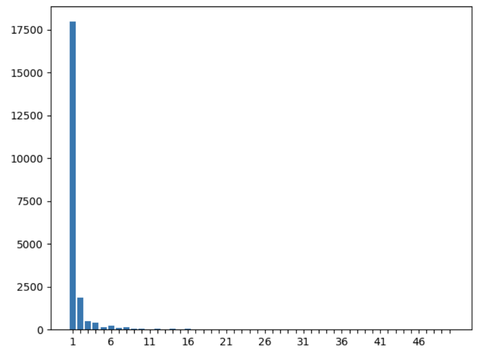


Figure 4: Distribution of Candidate Equation Number.

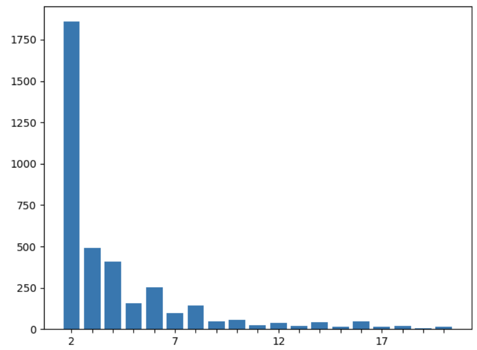


Figure 5: Distribution of Candidate Equation Number.

$$n![z^n]U^{\divideontimes}(z) \sim \frac{c_2\sqrt{r_1}}{n}(\frac{n}{r_2 e})^n \tag{30}$$

$$n![z^n]T^{\pm}(z) \sim \frac{c_3\sqrt{r_2}}{n}(\frac{n}{r_3 e})^n \tag{31}$$

$$n![z^n]T^{\divideontimes}(z) \sim \frac{c_4\sqrt{r_2}}{n}(\frac{n}{r_4 e})^n \tag{32}$$

So we have:

$$u_n = n![z^n]U(z) \sim \frac{(c_1 + c_2)\sqrt{r_1}}{n}(\frac{n}{r_1 e})^n \tag{33}$$

$$t_n = n![z^n]T(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}(\frac{n}{r_2 e})^n \tag{34}$$

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:

$$s_n = n![z^n]S(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}(\frac{n}{r_2 e})^n \tag{35}$$

Q.E.D.

B Distribution of Candidate Equations

The largest candidate equation number of one example is 3914. We show the distribution of candidate equations in Figure 4 and 5. The x-axis represents the number of candidates, while the y-axis represents the number of examples that have x candidate equations. We can see from Figure 4, 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 5, 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.

Algorithm 1 *enum_skel(n)*

```
Require: $n \geq 1$
  Initialize empty list skills
  for $i \leq n;\ i = 1;\ i++$ 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
```

C Experimental 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.

D ComSearch Details

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 $g(x)$. :

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

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 (Schröder, 1870), which calculates the number of labeled series-reduced rooted trees with n leaves. We use a deep-first search algorithm shown in Algorithm 1 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 $\mathcal{X} = \{x_k\}_{k=1}^{i}$ (Wang, 2021).

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 $\&$ 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 1 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.

$$unit\_skel_{div}(i) = \{(A\&\overline{A}) | A \subseteq \mathcal{X}; A \neq \emptyset\} \tag{36}$$

$$\begin{aligned} &unit\_skel_{sub}(i) = \\ &\quad \{((a(A - a))\&\overline{A - a}) | A \subseteq \mathcal{X}; a \in A\} \end{aligned} \tag{37}$$

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.